www.controltechniques.com



User Guide

Advanced Position Controller

Position Control for Unidrive SP

Part Number: 0471-0034-05
Issue Number: 5

General Information

Drive software version

This product is supplied with the latest version of user-interface and machine control software. If this product is to be used in a new or existing system with other drives, there may be some differences between their software and the software in this product. These differences may cause this product to function differently. This may also apply to drives returned from a Control Techniques Service Centre.

If there is any doubt, contact a Control Techniques Drive Centre.

Issue Code: 5

Contents

1 Using This Manual

This manual provides complete information on the setup and operation of the SM-Applications on board **A**dvanced **P**osition **C**ontroller. The manual has been written assuming that the user has the following knowledge:

- Setup and tuning of Unidrive SP
- Experience with programming SM-Applications with Sypt
- Motion applications experience

The sections within this manual are in a logical order, and will progress the user from basic safety information through to code examples. These sections are described in the table below:

Table 1-1

Section Name	Section Number	What the section is for
Safety Information	2	This section describes basic drive, and drive system safety
Introduction	3	This section gives simple over view of the APC's structure, background information, features, and hardware, together with additional basic information on the units used within the APC.
Functional Description	4	This section provides a detailed description of the various functions of the APC, like the references, what they do and how to set them up. Additional application information is also given.
APC Command Description	5	This section gives detailed information on each individual APC function command, and any other applicable commands.
Getting Started	6	This section shows how to get a very simple program working using the default settings. The user can construct a program from "building block" code examples given in this chapter.
Program Examples	7	This section provides several code examples, each with detailed notation, describing the action of the code. These examples may be copied and pasted into Sypt from the PDF version of this manual, to form the basis of new programs.
Application Notes	8	This section provides application notes on solving common industrial problems such as high load inertia overshoot.
Migration Guide And Software Changes	9	This Section Details the parameter changes introduced with SM-Applications firmware version >=V01.03.00, and the new features introduced with firmware version >=V01.03.02. It also gives information on common problems encountered when transferring code written for previous versions of the SM-Applications firmware.
Glossary of Terminology	10	This section gives definitions for general motion terminology, which is also used throughout this manual.
Quick Reference	11	This section contains short form description of all the command calls for the APC, read parameters, aliases, diagrams, menu 90 virtual parameters, and all the relevant SM-Applications parameters. This section is intended for users who are familiar with the APC to refer to whilst coding.

Text shown in *Italics* are cross reference links e.g. *APCSetRunMode()* is a link to that commands description in the APC Command Description section. These links can be used when the document is in PDF format.

Numbers shown in square brackets e.g. [28] are read parameter numbers, and can be used with the command *APCReadPar()* to return the value of that read parameter.

2 Safety Information

2.1 Warnings, Cautions and Notes

WARNING

A **Warning** contains information, which is essential for avoiding a safety hazard.

CAUTION

A **Caution** contains information, which is necessary for avoiding a risk of damage to the product or other equipment.

NOTE A **Note** contains information, which helps to ensure correct operation of the product.

2.2 Electrical Safety - General Warning

The voltages used in the drive can cause severe electrical shock and/or burns, and could be lethal. Extreme care is necessary at all times when working with or adjacent to the drive. Specific warnings are given at the relevant places in this User Guide.

2.3 System Design and Safety of Personnel

The drive is intended as a component for professional incorporation into complete equipment or a system. If installed incorrectly, the drive may present a safety hazard.

The drive uses high voltages and currents, carries a high level of stored electrical energy, and is used to control equipment which can cause injury.

Close attention is required to the electrical installation and the system design to avoid hazards either in normal operation or in the event of equipment malfunction. System design, installation, commissioning and maintenance must be carried out by personnel who have the necessary training and experience. They must read this safety information and this User Guide carefully.

The STOP and SECURE DISABLE functions of the drive do not isolate dangerous voltages from the output of the drive or from any external option unit. The supply must be disconnected by an approved electrical isolation device before gaining access to the electrical connections.

With the sole exception of the SECURE DISABLE function, none of the drive functions must be used to ensure safety of personnel, i.e. they must not be used for safety-related functions.

Careful consideration must be given to the functions of the drive which might result in a hazard, either through their intended behavior or through incorrect operation due to a fault. In any application where a malfunction of the drive or its control system could lead to or allow damage, loss or injury, a risk analysis must be carried out, and where necessary, further measures taken to reduce the risk - for example, an over-speed protection device in case of failure of the speed control, or a fail-safe mechanical brake in case of loss of motor braking.

The SECURE DISABLE function has been approved[1] as meeting the requirements of EN954-1 category 3 for the prevention of unexpected starting of the drive. It may be used in a safety-related application. **The system designer is responsible for ensuring that the complete system is safe and designed correctly according to the relevant safety standards.**

[1]Independent approval by BIA has been given for sizes 1 to 3.

2.4 Environmental Limits

Instructions in the *Unidrive SP User Guide* regarding transport, storage, installation and use of the drive must be complied with, including the specified environmental limits. Drives must not be subjected to excessive physical force.

2.5 Compliance with Regulations

The installer is responsible for complying with all relevant regulations, such as national wiring regulations, accident prevention regulations and electromagnetic compatibility (EMC) regulations. Particular attention must be given to the cross-sectional areas of conductors, the selection of fuses or other protection, and protective earth (ground) connections.

The *Unidrive SP User Guide* contains instruction for achieving compliance with specific EMC standards.

Within the European Union, all machinery in which this product is used must comply with the following directives:

98/37/EC: Safety of machinery.

89/336/EEC: Electromagnetic Compatibility.

2.6 Motor

Ensure the motor is installed in accordance with the manufacturer's recommendations. Ensure the motor shaft is not exposed.

Standard squirrel cage induction motors are designed for single speed operation. If it is intended to use the capability of the drive to run a motor at speeds above its designed maximum, it is strongly recommended that the manufacturer is consulted first.

Low speeds may cause the motor to overheat because the cooling fan becomes less effective. The motor should be fitted with a protection thermistor. If necessary, an electric forced vent fan should be used.

The values of the motor parameters set in the drive affect the protection of the motor. The default values in the drive should not be relied upon.

It is essential that the correct value is entered in Pr **0.46** motor rated current. This affects the thermal protection of the motor.

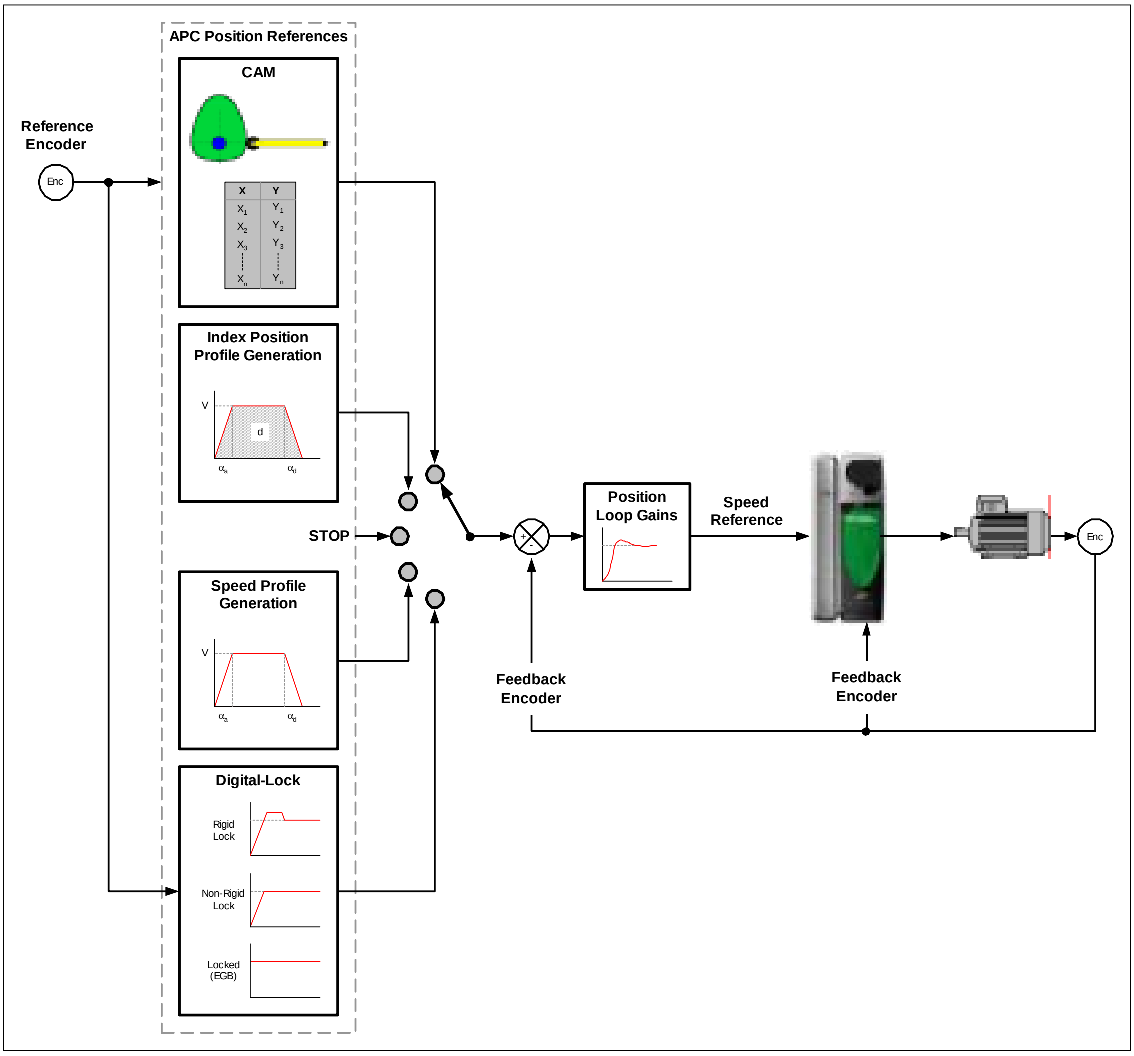
2.7 Adjusting Parameters

Some parameters have a profound effect on the operation of the drive. They must not be altered without careful consideration of the impact on the controlled system. Measures must be taken to prevent unwanted changes due to error or tampering.

3 Introduction

The **A**dvanced **P**osition **C**ontroller is a multifunction motion kernel, which is embedded into the SM-Applications module operating system. It has very similar features to its predecessor on the UD70, with some enhancements relating to the user interface, functionality and efficiency. The APC cannot be used on its own as it requires user code for sequencing, scaling and control of the motion functionality contained within the APC kernel.

Figure 3-1



The APC with Unidrive SP, can be used in a wide range of closed loop applications, in Servo or Closed Loop Vector modes. Rotational and linear motors are supported with a wide range of feedback devices e.g. SinCos, Incremental Quadrature, EndAt and SSI.

The APC Controls the motion of one drive axis, and can be used to:

- Perform independent discrete positional moves, where the motion trajectory is produced by a profile generator, which controls acceleration, deceleration and maximum speed. Typical applications include pushers, feeders, indexers etc.
- Produce synchronised motion with respect to another axis. Typical uses include simple following applications like Digital Lock or Electronic Gear Box (EGB), or more complex synchronised profiles using an electronic CAM e.g. Flying Shear and Rotary Knife etc.

3.1 Overview

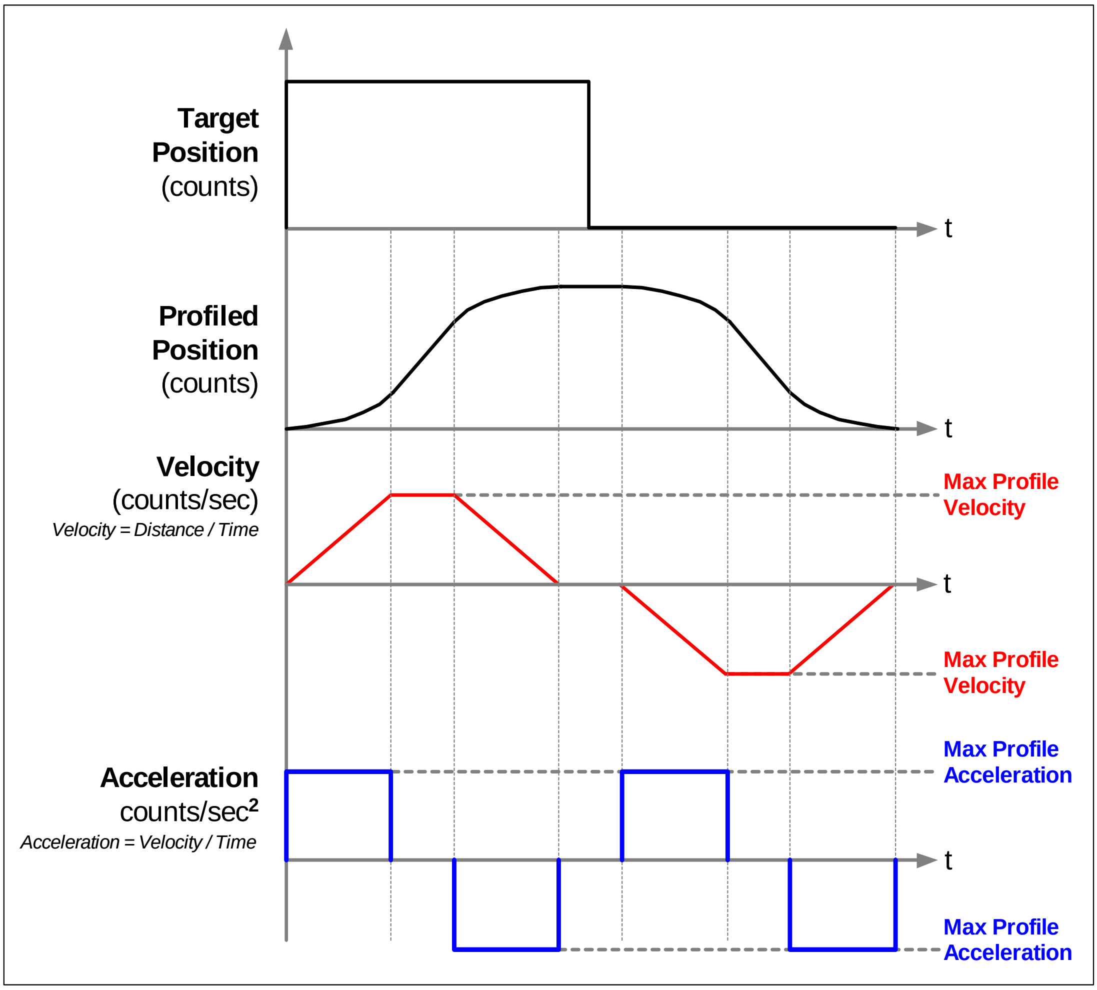
3.1.1 Features

1. There are 5 motion function references, to meet a wide range of applications:
   - Stop
   - Position
   - Speed
   - CAM
     - Multiple interpolation
     - Single shot or cyclic modes
     - Recovery of CAM position after power loss
   - Digital Lock
     - Rigid and Non-Rigid functionality
     - Numerator/Denominator Ratio
2. Position Profile generator, which enables the user to change any of its parameters and take immediate effect during a profile
3. Offset Profile generator, which enables the user to add a separate speed or position offset to any of the APC references (except the Stop reference).
4. Bump-less transition when selecting references.
5. Wide range of encoder interfaces, absolute and incremental.
6. User definable position resolution (within the limitations of the encoder)
7. Marker pulse and Freeze Capture
8. Flexible and open interface for user program
9. At Speed and At Position flags for the Main and Offset profile generators
10. The ability to insert filters or other function blocks after the source counters for both the Reference and Feedback position counters.
11. Opearation with user units values e.g. mm, mm/s, mm/s/s

3.2 Motion Fundamentals

Figure 3-2 shows the relationship of the position, velocity and acceleration, when applied to a trapezoidal velocity profile.

Figure 3-2



3.3 Clarification of Terms

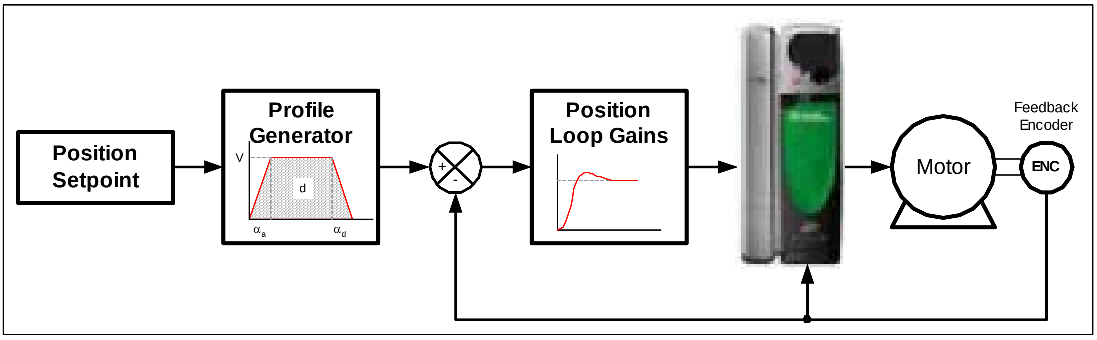
Throughout this document the terminology shown in Table 3-1 will be used:

Table 3-1

Term	Single Axis	1.5 Axes
Reference	Set point for motion loop	This is the Master or the Auxiliary axis
Feedback	Main feedback to motion loop	This is the Slave or the Main drive axis

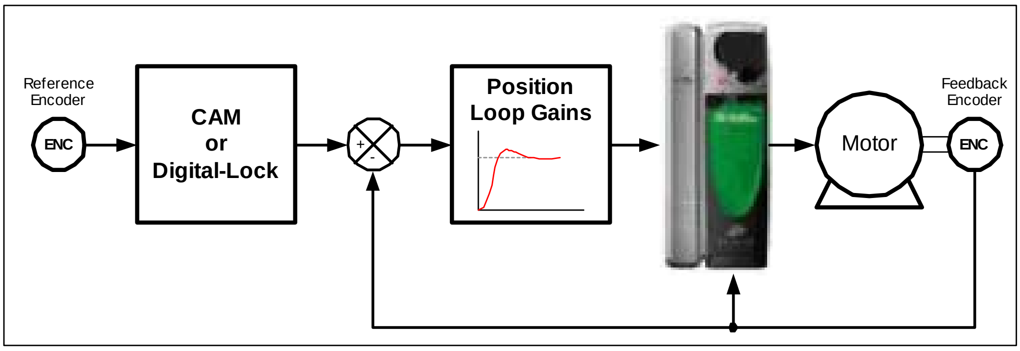
- Single axis means one stand alone controllable axis, where the set point is not directly related to another axis' position/speed. See Figure 3-3 below:

Figure 3-3



- 1.5 axis means there is one controllable axis following another axis synchronously in position/speed. See Figure 3-4:

Figure 3-4



3.4 Positional Feedback/References

The reference and feedback positions can be taken from a feedback device connected directly to the drive, a feedback device connected to a position category option module in any drive slot, or from a user selected parameter. The latter option can be used in conjunction with CTSync to implement a virtual master system. The update rates associated with these options are shown in Table 3-2 below:

Table 3-2

Reference		
Source	**Comment**	**Update Time**
Unidrive-SP (D-Type)	Supports Quadrature, F&D, CW/CCW, SinCos & SSI	250µs min.
SM-Universal Encoder Plus	Supports Quadrature, F&D, CW/CCW, SinCos & SSI	250µs min.
SM-Encoder Plus	Supports Quadrature, F&D, CW/CCW	4ms min.
SM-Resolver	Supports Resolver only	4ms min.
User Parameter / CT-Sync	Via SM-Application port RS485	250µs min.
Feedback		
Source	**Comment**	**Update Time**
Unidrive-SP (D-Type)	Supports Quadrature, F&D, CW/CCW, SinCos & SSI	250µs min.
SM-Universal Encoder Plus	Supports Quadrature, F&D, CW/CCW, SinCos & SSI	250µs min.
SM-Encoder Plus	Supports Quadrature, F&D, CW/CCW	250µs min.
SM-Resolver	Supports Resolver only	250µs min.
User Parameter / CT-Sync	Via SM-Application port RS485	250µs min.

Refer to the options relevant user manual for more information.

NOTE The initialisation time for absolute encoders may be in excess of 300ms, so to make sure the encoders have been properly initialised the user should check drive parameter #3.48 = 1 in the Initial task before allowing any motion code to run.

3.5 Positioning Modes

The APC reference and feedback position counters are incremented by integrating the encoder counts per sample. The start or reset position can be defined by one of the following modes: -

3.5.1 Relative Mode

In this mode the reference and feedback counters are set to zero when an APC Reset is actioned. If required an offset position can be added to the feedback counters.

Any encoder can be used in this mode.

3.5.2 Absolute Mode

In this mode the reference and the feedback counters are set to the absolute position values read directly from the source encoder when an APC Reset is actioned. If required an offset position can be added to the feedback counters.

This is useful when using absolute feedback devices like SSI, SinCos, EndAt, as the absolute position is maintained at all times, even without power.

3.6 Resolution

3.6.1 The APC kernel resolution

The APC uses 32 bit signed resolution.

3.6.2 Encoder Resolution

3.6.2.1 Scaling

The reference and feedback encoder position counters can be scaled from 2^{16} to 2^{31} depending on the encoder used. For example, high resolution SinCos encoder can have a maximum of 2^{22}bit resolution per turn, which can be scaled down to a minimum of 2^{16}.

When the resolution of the encoder counts per revolution is less than 2^{16}, the encoder counts will be interpolated up to 2^{16}.

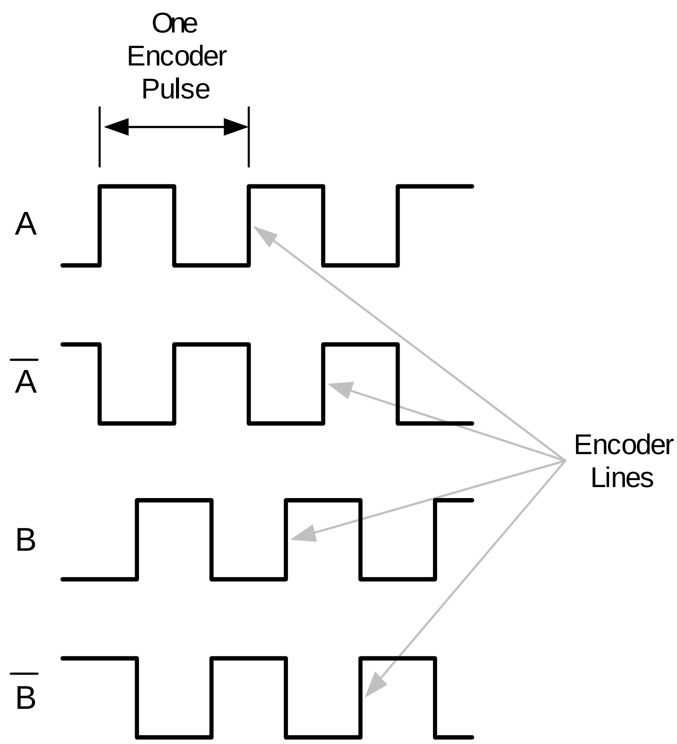
Note: The true resolution will still equal the encoder counts per revolution, though it has been interpolated up to 2^{16}, each count will go up in large steps.

E.g. for 1024ppr encoder the counts per rev will be 4096. For one count the counter will go up in steps of 16 (65536/4096).

3.6.2.2 Incremental Encoder Resolution

Figure 3-5 shows how to determine the number of encoder lines, (or counts per revolution) for a quadrature incremental encoder.

Figure 3-5



The number of lines per revolution = 4 * encoder pulses per revolution.

3.6.2.3 SinCos Encoder Feedback Resolution.

When operating with SinCos Encoder feedback the maximum resolution is determined from:

1. The number of sine waves of the encoder.
2. The interpolated information resolution.

The Max. Feedback Resolution per turn = Number of sine waves * Interpolated resolution

The interpolated information resolution can vary between a maximum of 11 bits (2048) to a minimum of 6 bits (64) with this being determined from

1. The input frequency.
2. The encoder voltage levels.

Drive encoder input Table 3-3:

Table 3-3

Feedback Signal Voltage Level	Feedback Signal Frequency 1kHz	5kHz	50kHz	100kHz	200kHz	500kHz
1.2Vdc	2048	2048	1024	1024	512	256
1.0Vdc	2048	2048	1024	512	512	128
0.8Vdc	1024	1024	1024	512	256	128
0.6Vdc	1024	1024	512	512	256	128
0.4Vdc	512	512	512	256	128	64

SM-Universal Encoder Plus Encoder input Table 3-4.

Table 3-4

Feedback Signal Voltage Level	Feedback Signal Frequency				
	1kHz	**5kHz**	**50kHz**	**100kHz**	**166kHz**
1.2Vdc	2048	2048	1024	1024	512
1.0Vdc	2048	2048	1024	512	512
0.8Vdc	1024	1024	1024	512	256
0.6Vdc	1024	1024	512	512	256
0.4Vdc	512	512	512	256	128

3.6.2.4 Synchronous Serial Interface (SSI) and EndAt Encoder Resolution

The resolution of this encoder is given as number of turn bits and the number of bits per turn. For example an SSI or EndAt encoder may be described as 25bit, where 13bits are the number of turn bits and the 12bits are for the number of counts per revolution. In this case the counts will be interpolated up to 16bits by the drive, so each increment will be in 16 count steps.

NOTE Please Refer to the relevant user manual for more detailed information.

3.7 Hardware

To utilise the Advanced Position Controller the following products are used:

- Unidrive SP
- SM-Applications module or SM-Applications Lite module

3.7.1 SM-Applications Module

The SM-Applications module for Unidrive SP is an option module that can be fitted to any one of the three expansion slots in the Unidrive SP and is powered from the Unidrive SP internal power supply.

3.7.1.1 Specifications

- High speed dedicated microprocessor
- 384kb Flash memory for user program
- 80kb user program memory
- EIA-RS485 port offering ANSI, Modbus-RTU slave and master, and Modbus-ASCII slave and master protocols
- CTNet high speed network connection offering up to 5Mbit/s data rate
- Two high speed 24V digital inputs and outputs
- Dual-port RAM interface for communicating with the Unidrive SP and other option modules

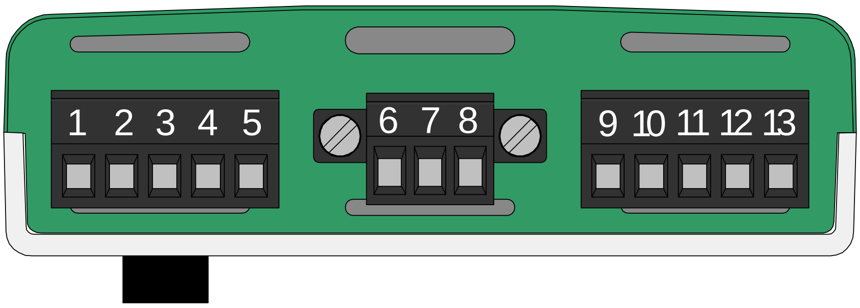
Task based programming system allowing for real-time control of drive and process.

This is a general overview only. For more information refer to the SM-Applications manual for more information.

3.7.1.2 Electrical Connections

The SM-Applications module (Figure 3-6) has 2 no. 5-way and 1 no. 3-way screw terminal blocks.

Figure 3-6



The terminals are numbered from terminal 1 on the left hand side to terminal 13 on the right. The terminal functions are given in Table 3-5 below:

Table 3-5

Terminal	Function	Description
1	0V SC	0V connection for RS485 port
2	/RX	EIA-RS485 Receive line (negative). Incoming.
3	RX	EIA-RS485 Receive line (positive). Incoming.
4	/TX	EIA-RS485 Transmit line (negative). Outgoing.
5	TX	EIA-RS485 Transmit line (positive). Outgoing.
6	CTNet A	CTNet data line
7	CTNet Shield	Shield connection for CTNet
8	CTNet B	CTNet data line
9	0V	0V connection for digital I/O
10	DI0	Digital input 0 (optional Freeze input)
11	DI1	Digital input 1
12	DO0	Digital output 0
13	DO1	Digital output 1

3.7.2 SM-Applications Lite module

The SM-Applications Lite module is a lower-cost version of the SM-Applications module. Therefore it does not have some of the features of the fully featured module. These include:

- <384kb Flash memory for user program (384kb for SM-Applications).
- <80kb user program memory (80kb for SM-Applications).
- No terminal connections - no RS485 port, no CTNet port and no Digital I/O.

The user cannot use CTSync or freeze position data using the SM-Applications Lite module, due to it not having the RS485 port or digital I/O. To get freeze position data when an SM-Applications Lite module is used, a Universal Encoder Plus module is required.

3.8 High Speed Position Capture

3.8.1 Marker Pulse

Discrete marker pulse capture position facilities are available for reference and Feedback incremental encoders. With the correct interfacing hardware, (24Vdc to RS485 converter), the marker pulse can be utilised as an additional high speed Freeze/ registration input.

3.8.2 Freeze

A high-speed freeze input can be used to capture the Drive (D-Type), encoder position, or both the Drive encoder position and SM option position together. The Freeze hardware can be configured to capture on a positive or negative edge in both the SM-Applications and SM-Universal Encoder Plus. Table 3-6 below shows the combinations available with the SM option used.

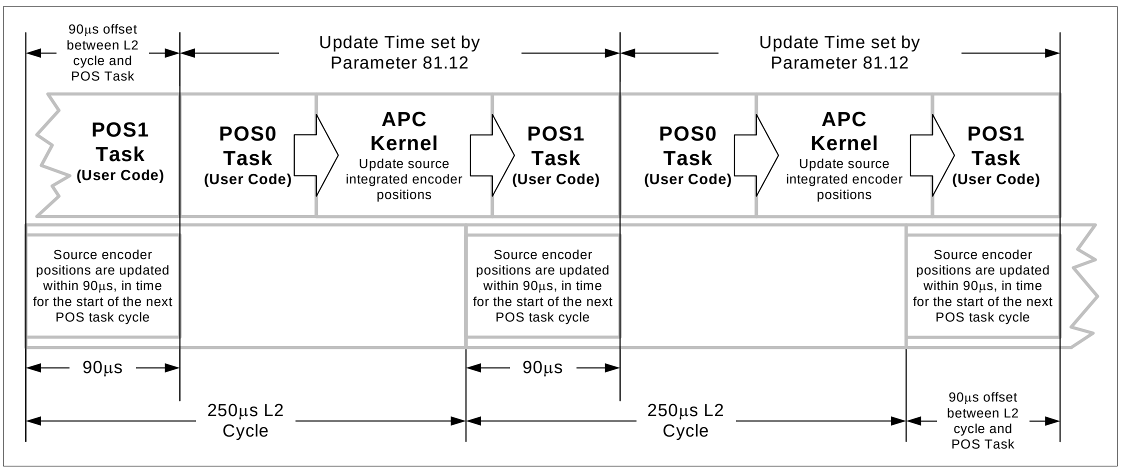
Table 3-6

SM-Option Module	Input & Signal Type	Captured Positions
SM-Applications	(DIGIN0) T10 - 24Vdc	Drive (D-Type) Position SM-Encoder Plus SM-Resolver
SM-Universal Encoder Plus	(PL2) T1 - 24Vdc T8/T9 - RS485	Drive (D-Type) Position SM-Encoder Plus SM-Resolver SM-Universal Encoder Plus

3.9 Task model

The APC kernel is executed between the POS0 and POS1 tasks of the SM-Applications operating system, therefore allowing the user write code before and after the APC runs. The POS task cycle shown below, is updated at a rate given by Pr81.12. Figure 3-7 shows the POS task cycle.

Figure 3-7



3.10 Performance

3.10.1 Update Rates

The motion/position task can be runn at the following update rates and is set by parameter 81.12:

250µs, 500µs, 1ms, 2ms, 4ms & 8ms.

To run the APC effectively and reliability with user code it is recommended to set the update rate no faster than 500µs. Some functions will run at 250µs, but there will be limited resources available for user code, and for CTNet to run.

The I/O on the Drive and SM-Applications module can be read at either 250µs or the POS task update rate from the following parameters:

- Virtual Parameter 91.16 for the Drive I/O

Parameter 86.01 to 86.04 for the SM Applications I/O.

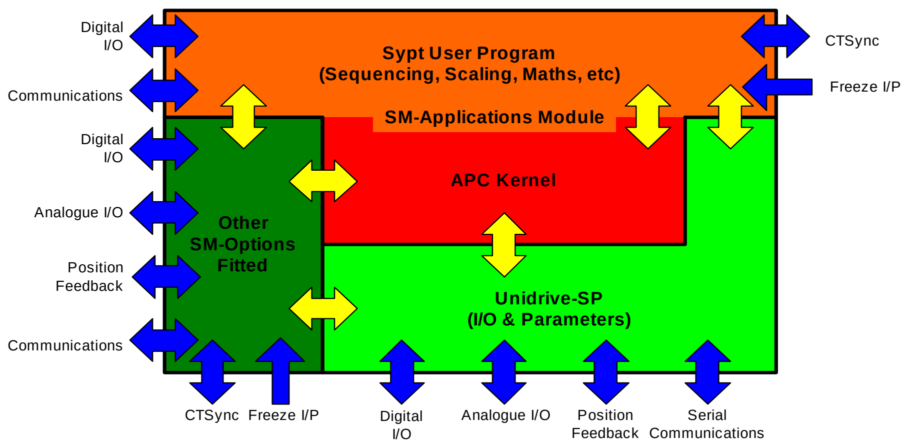
NOTE If the Drive I/O is read from the Drive I/O parameters, parameters 8.01 to 8.06 and 8.09, the update rate is only 4ms.

3.11 Interfacing

3.11.1 Overview

Figure 3-8 shows an overview of the interfacing between the SM-Options, user program, Unidrive SP, and the APC.

Figure 3-8



The user code can access any parameter, I/O or Option module used on Unidrive SP.

3.11.2 Units

The APC Kernel uses the following units to maintain maximum resolution of each parameter:

3.11.2.1 Position

The position is set in Encoder counts, so the resolution will correspond to set feedback encoder resolution. For example:

- **Rotational** - If there are 65536 counts per turn, to move 5 revolutions, the position reference will be 5 x 65536 = 327680.
- **Linear** - If there are 2048 counts per mm, to move 100mm the APC position reference has to be 2048 x 100 = 204800.

Please refer to the *Resolution* section for more information on setpoint resolution.

3.11.2.2 Speed

The APC internal speed units are in 2^{32} encoder counts per rev over a 250µs sample. These units are used for the Profile Maximum Speed, Speed Reference, Reference and Feedback Encoder Speed, Profile Generator Output Speed, and the APC Output Speed Reference. The following equations convert rpm into internal speed units:

APCSpeed = rpm * APC Maximum Speed / 60 (revs/s) * 4000 (revs/250us)

APC Speed = rpm * 2^{32} / 60 * 4000

or

APC Speed = rpm * 2^{32} / 240000 = rpm * 2^{31} / 120000 = rpm * 2^{30} / 60000

For SM-Applications programs use the following equation:

`APCSpeed%=MULDIV(SpeedRPM%, 1073741824, 60000)` = rpm * 2^{30} /60000

3.11.2.3 Acceleration

The APC internal acceleration units are in 2^{32} encoder counts per rev over 250μs / 250μs. These units are used for the Profile Acceleration rate, Profile Deceleration rate, and the Profile Output Acceleration.

APC Acceleration rate =

APC Speed (2^{32} encoder counts per rev over 250μs) * 250μs / Accel. Time(μs)

For SM-Applications programs use the following equation:

```
APCAccel%=MULDIV(APCSpeed%,250,AccelTimeMicroSec%)
```

3.11.2.4 Proportional Gain (Kv)

The APC internal Proportional gain units are in 0.01rads/s / rad units. If required, the word “rad” or “rads” in the proportional gain units, can substituted for any other units like mm or revs, as the response will still be the same.

3.11.3 User Program

The interface to the APC is not via parameters, but by a series of function calls to the main APC kernel. In most cases the operations read from, or written, to a parameter that is held within the kernel. None of the data held by the kernel can be saved directly into non-volatile memory, and so the user program must provide this function. The function calls are described later in this manual in the *APC Command Descriptions* section. The Sypt Pro programming tool is required to call and configure the APC motion kernel.

User Programs can be downloaded from the Sypt Pro programming tool to the SM-Applications module by the following methods:

- RS 485, using the Unidrive SP’s RJ45 comms port on the front of the drive. A converter from RS232 or USB, to RS485 may be required. This is the only method which can be used to down load to an SM-Applications Lite option module.
- CTNet, via the dedicated terminals on the SM-Applications module. This is the fastest way to transfer programs. A CTNet card will be required to transfer the program from the user PC to the SM-Application module.

The APC command calls are worded descriptions of their function, which means that they can be time consuming to enter in to DPL code. To save time and spelling mistakes, use the Insert Function Block button , or select Insert Function Block from the Insert menu in the DPL editor.

This page is intentionally blank.

4 Functional Description

4.1 APC Overall Diagram

Figure 4-1

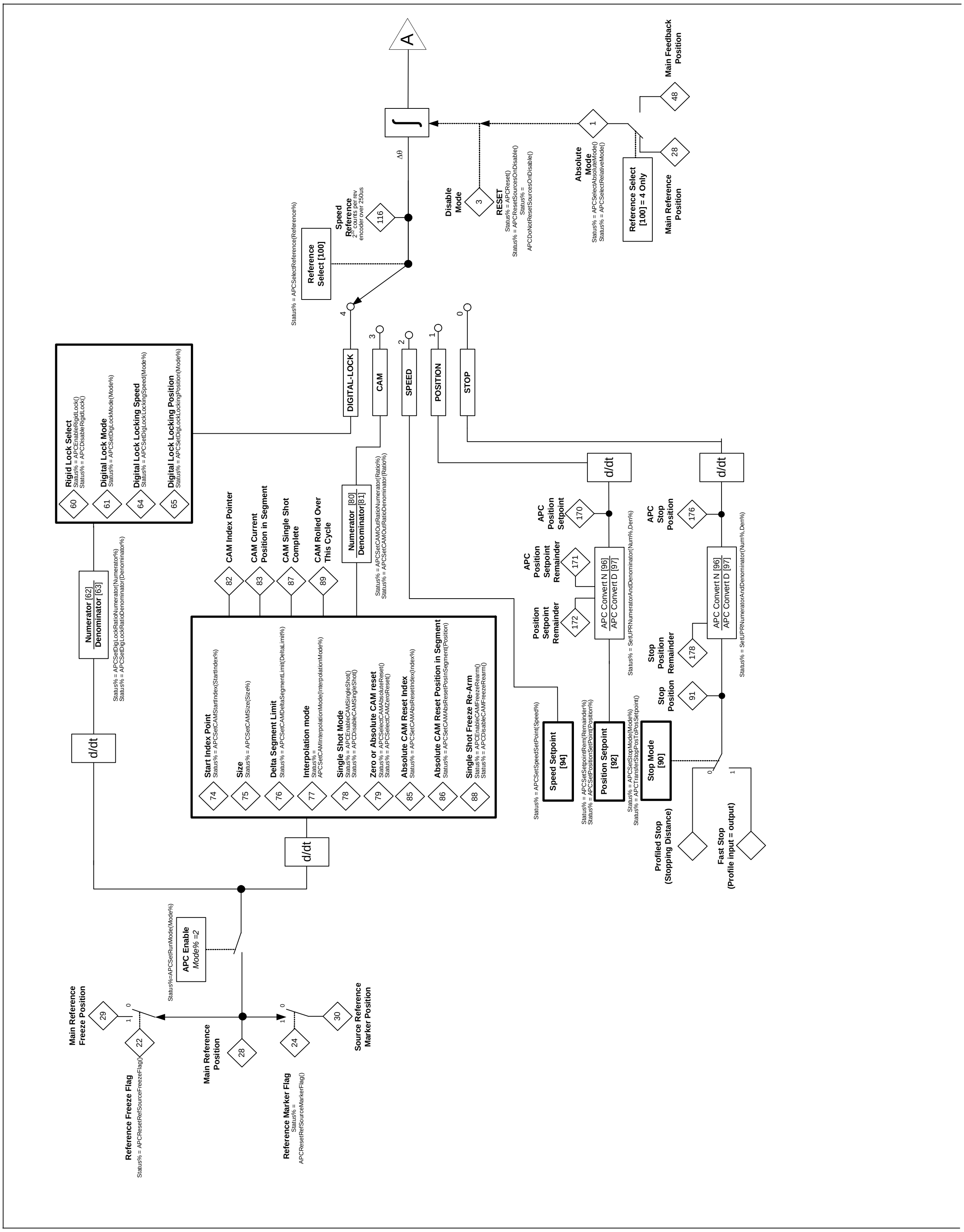
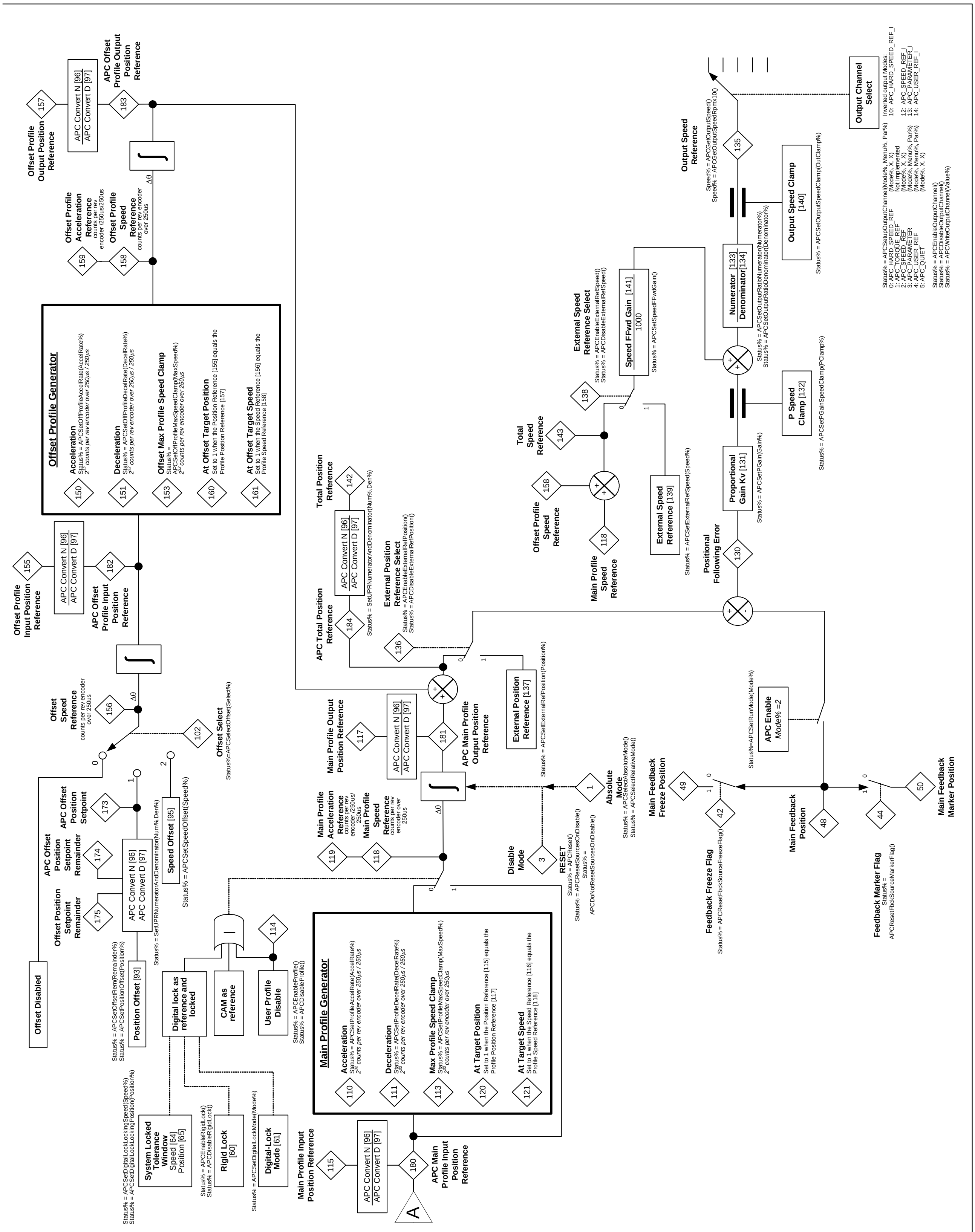


Figure 4-2


Offset Profile Generator
Main Profile Generator
Output Channel Select
Offset Disabled
Position Offset [93]
Speed Offset [95]
External Speed Reference [139]
External Position Reference [137]
Output Speed Clamp [140]
P Speed Clamp [132]
Proportional Gain Kv [131]
APC Enable Mode% =2

4.2 Operational Overview

4.2.1 Modes of Operation

The APC can be configured to run in 3 modes:

1. Off - In this mode the APC is never called
2. Disabled - In this mode only the encoder position counters and the following error are updated. The encoders selected will also configure the corresponding virtual parameters in menu 90.
3. Enabled - In this mode the complete APC functionality is available.

4.2.2 Reference Switching

There are 5 operating references available with the APC:

- Stop
- Position
- Speed
- CAM
- Digital Lock

The user may switch between each of the references, with “bump-less” transfer, i.e. the profile generator is used to ramp to the new position or speed reference, so that there are no sudden jumps in Feedback axis position or speed, with the following exceptions:

- Selecting the CAM reference when the Feedback axis is not at standstill
- Selecting the Stop reference when Instant Stop is selected, and the Feedback Axis is not at standstill.

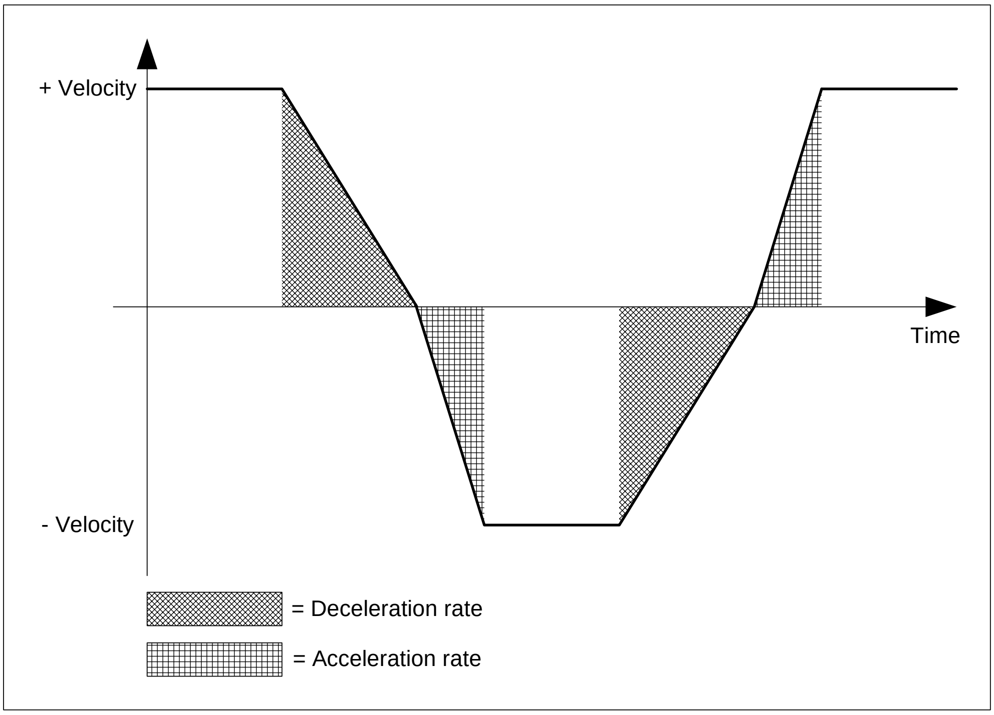
4.2.3 Profile Generators

The Profile Generators (main and offset) determine how a change in speed or position reference is responded to. The Profile Generators operate differently for a change in speed reference, compared with a change in position reference. These differences are discussed below:

- For a change in speed reference, the profile generators calculate a ramp to the new speed reference, with respect to the acceleration / deceleration rate set.
- For a change in position reference, the profile generator calculates the time required at maximum speed, with respect to the acceleration / deceleration rate set, so that the feedback axis will ramp down and stop at, or synchronise with the new position reference

When the main Profile generator is inactive, any change in speed or position reference will be responded to instantly with no ramps. The maximum speed reached is still controlled by the position loop output speed clamp.

Figure shows how both profile generators acceleration and deceleration rates are implemented:



Both Profile Generators have At Target Position[120] [160], and At Target Speed [121] [161] indicators. The flags are high when profile input position or speed equals profile output speed or position respectively.

4.2.3.1 Offset Profile Generator

The Offset Profile generator has two selectable input references; a position and a speed offset. Only one of these references may be used at a time.

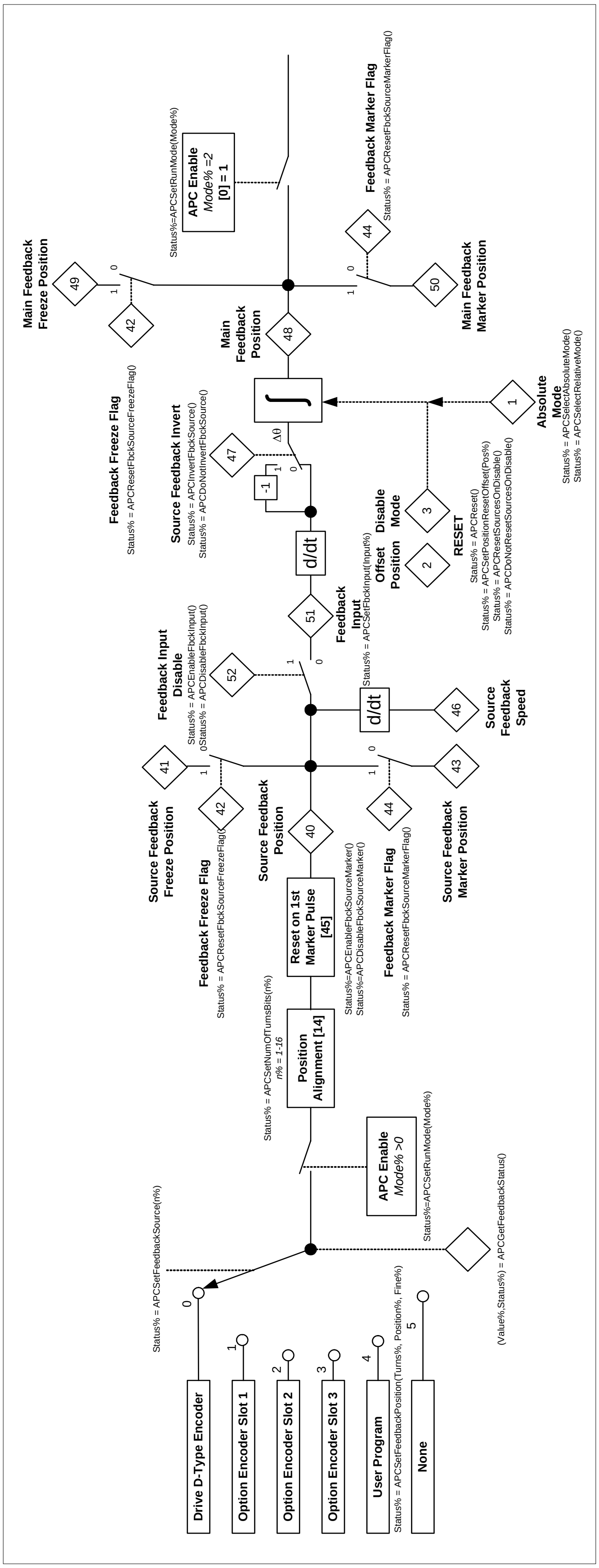
If the Offset Speed and Position references are selected from one another i.e. switched from Speed to Position, or position to speed, then the Offset Profile accel and decel rates will be used. However, If the Speed or Position offset is selected, and then changed to offsets disabled, the output of the Offset Profile Generator is set to 0, and therefore the Main Profile Generator accel and decel rate will be used to profile back to the demand speed or position.

The offset profile decel rate is not used when decelerating to a standstill, when the Stop reference, and Profiled Stop, [90] = 0, is selected. Under these conditions, the main profile generator decel rate is used to decelerate the feedback axis.

4.3 Reference and Feedback Encoder Positions

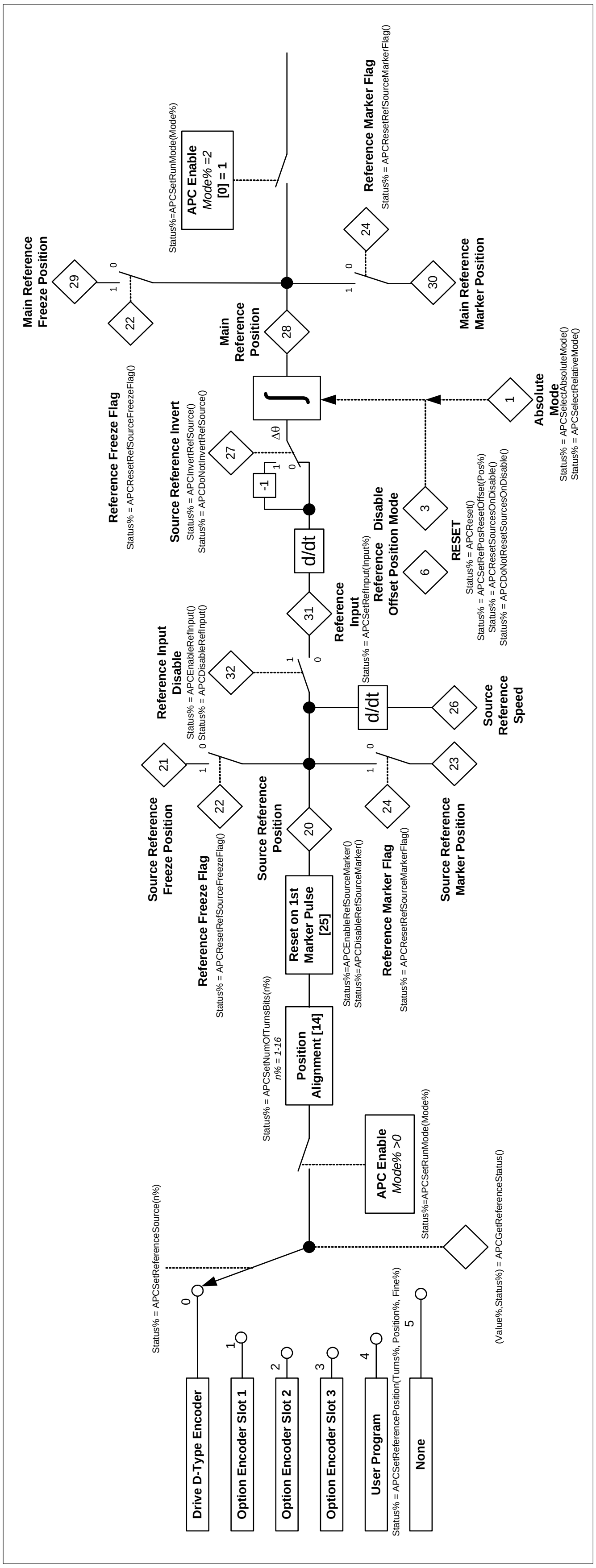
Feedback encoder position counter (Slave):

Figure 4-3



Reference encoder position counter (Master):

Figure 4-4



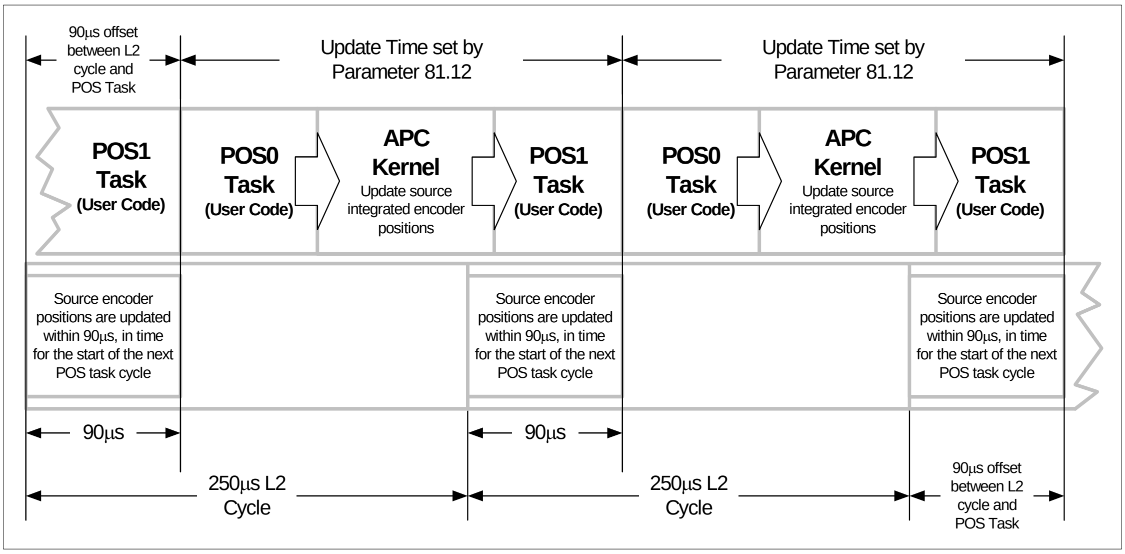
4.3.1 Enabling the APC

When the APC is enabled using APCSetRunMode(2), the APC will run in disabled for 1 position task sample in order to update the source encoder counters. This means that every time the run mode is changed from 0 or 1 to 2, the user must either wait for one POS task cycle to pass, or check that the APC is enabled before using the counter values. To make this easy, read parameter [4] has been introduced to allow the user to check; 0 = disabled 1 = enabled. Read parameter [4] differs from [0] in that read parameter 0 is set before the APC Runs and indicates a user request to disable or enable, whereas read parameter [4] is set at the end of the APC kernel run time, and indicated that the request has been accepted and actioned.

4.3.2 Encoder Source Update.

Using the APC the encoder source positions are updated before the POS0 task runs as shown by Figure 4-5 below:

Figure 4-5



When using the APC in Disabled mode, where only the encoders are used, the source encoder positions can be updated at a rate defined by parameter 81.16, this is as follows:

0 = 250µs (every speed loop update)

1 = At the start of every POS task (Defined by 81.12)

2 = At the start of every Clock task (Defined by 81.11)

3 = Not Updated

When using the APC for best performance 81.16 should be set to 1.

4.3.3 Encoder Source Selection and Main Position Integrators

Any encoder that is connected to either the Unidrive SP or SM-Position option, can be used as the APC Reference or Feedback source.

The SM-Applications virtual parameters (menu 90), will be setup with respect to the APC Reference and Feedback integrated counters are incremented/decremented with the delta position per sample of the drive encoder positions. This allows the counters to be easily reset/preset in absolute or relative mode.

The source position can be inverted to change the direction of rotation.

4.3.4 Absolute and Relative Modes

The internal integrator counters for the Reference feedback can be used in:

- **Absolute mode** - where the internal integrated position counters are preset to the drive position counters (+ offset), on an APC reset. This is useful where absolute feedback devices are used. Definite datum position is known.
- **Relative mode** - where the internal counters are reset to 0 position (+Offset), on a APC reset where it is up to the user to define the datum position.

4.3.5 Resetting Internal Counters

The counters can be reset with a defined offset, if required, in 2 ways

1. When the APC is disabled, the APC can be set-up to reset the counters.
2. By using the reset command, *APCReset().* This can be used without disabling the APC.

NOTE Offsets are only added to feedback integrators.

4.3.6 Overview of Resetting Internal Counters

Table 4-1 indicates the reset values for each of the position integrators, used within the APC:

Table 4-1

Reference selected	Mode	Reference Integrator	Feedback Integrator	Profile Generator I/P	Profile Generator O/P
Stop	Relative	0	0*	Feedback Integrator	Feedback Integrator
	Absolute	Reference Source Position	Feedback Source Position	Feedback Source Position	Feedback Integrator
Position	Relative	0	0*	Feedback Integrator	Feedback Integrator
	Absolute	Reference Source Position	Feedback Source Position	Feedback Integrator	Feedback Integrator
Speed	Relative	0	0*	Feedback Integrator	Feedback Integrator
	Absolute	Reference Source Position	Feedback Source Position	Feedback Integrator	Feedback Integrator
CAM	Relative	0	0*	Feedback Integrator	Feedback Integrator
	Absolute	Reference Source Position	Feedback Source Position	Feedback Integrator	Feedback Integrator
Digital Lock	Relative	0	0*	Reference Integrator	Feedback Integrator
	Absolute	Reference Source Position	Feedback Source Position	Reference Integrator	Feedback Integrator

*Denotes reset position offset will be added to these values.

4.3.7 Resolution Alignment

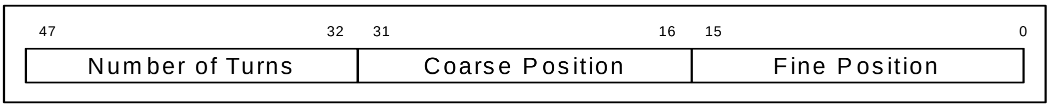
The drive uses 48bit counters for all the source encoder positions used on Unidrive-SP, and is allocated as turns, position and fine.

Encoders with less than 16bit counts per turn are interpolated up to 16bit, and will use the upper 32bits of the 48bit counter. The overall resolution will not be greater than the source encoder, as the counter will increment/decrement in larger steps.

> e.g. For a 4096ppr encoder there are 16384cpr, which means that after interpolation, the source counter will count in 4 count steps.

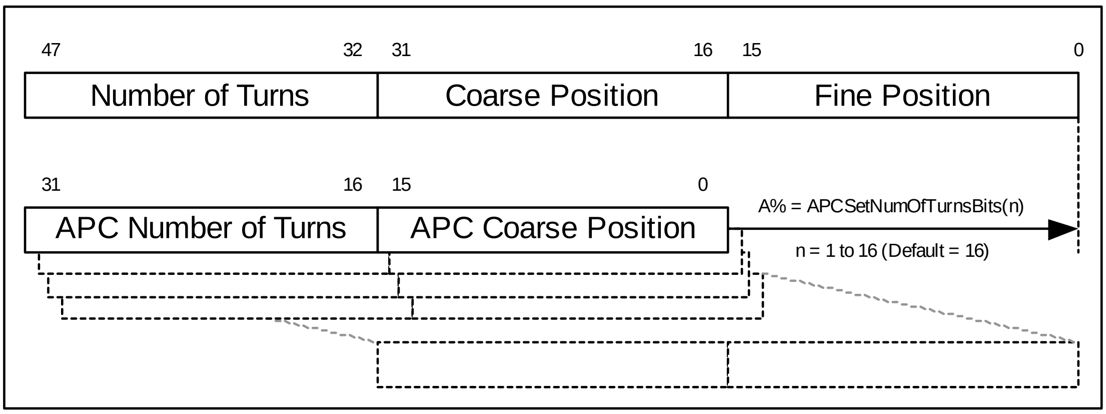
If higher resolution encoders are used the lower 16bits (fine) are also used. Figure 4-6 shows the construction of the 48bit position counters:

Figure 4-6



The APC/SM-Applications uses 32bit signed integer values, so on higher resolution encoders e.g. SinCos, the user can define the resolution per turn required, by defining the number of turn bits. See Figure 4-7:

Figure 4-7



4.3.8 Freeze and Marker pulse Functionality

The APC Feedback and Reference Source counters have Freeze and marker pulse functionality. This functionality includes:

- Freeze and Marker Pulse position capture
- Marker position reset (only whilst marker position capture is not used)
- Freeze Triggering of APC References (CAM and Digital Lock)

4.3.8.1 Freeze and Marker Pulse Position Capture

The APC source counters are equipped with 2 pairs of 32bit signed position single shot counters. Each pair of counters consists of an integrated and non-integrated position counter, both triggered simultaneously from one trigger source, one pair by the Freeze input (present on SM-Universal Encoder Plus and SM-Applications), and one pair by Marker Pulse.

The command *APCReadPar* must be used to read the captured positions. See Table 4-2:

Table 4-2

Position Counter	Command
Reference Freeze Non-Integrated [21]	*(Value%)=APCReadPar(21)*
Reference Freeze Integrated [29]	*(Value%)=APCReadPar(29)*
Reference Marker Non-Integrated [23]	*(Value%)=APCReadPar(23)*
Reference Marker Integrated [30]	*(Value%)=APCReadPar(30)*
Feedback Freeze Non-Integrated [41]	*(Value%)=APCReadPar(41)*
Feedback Marker Integrated [49]	*(Value%)=APCReadPar(49)*
Feedback Marker Non-Integrated [43]	*(Value%)=APCReadPar(43)*
Feedback Marker Integrated [50]	*(Value%)=APCReadPar(50)*

The actual position is captured within a micro second on the drive, and is passed to the SM-Applications module on the following sample period, detailed in Table 4-3:

Table 4-3

Source	Transfer to APC Time
Freeze	Every POS task
Marker (Drive Software V01.05.00)	Every 4ms
Marker (Drive Software V01.06.00)	Every POS task, but the marker position capture repetition period cannot exceed 20ms.

4.3.8.2 Freeze Position Capture

To action a Freeze capture, the following commands / virtual parameters shown in Table 4-4 have to be set:

Table 4-4

Parameter / Command	Function
APCEnableReferenceFreeze()	Reference Freeze Enable
APCEnableFeedbackFreeze()	Feedback Freeze Enable
APCResetRefSourceFreezeFlag()	Resets Reference Freeze flag
APCResetFbckSourceFreezeFlag()	Resets Feedback Freeze flag

The Freeze hardware also has to be setup, however this is dependent on which option modules have been used for the Feedback and Reference source. Table 4-5 shows which parameters have to be set for the freeze hardware, for each configuration of option modules:

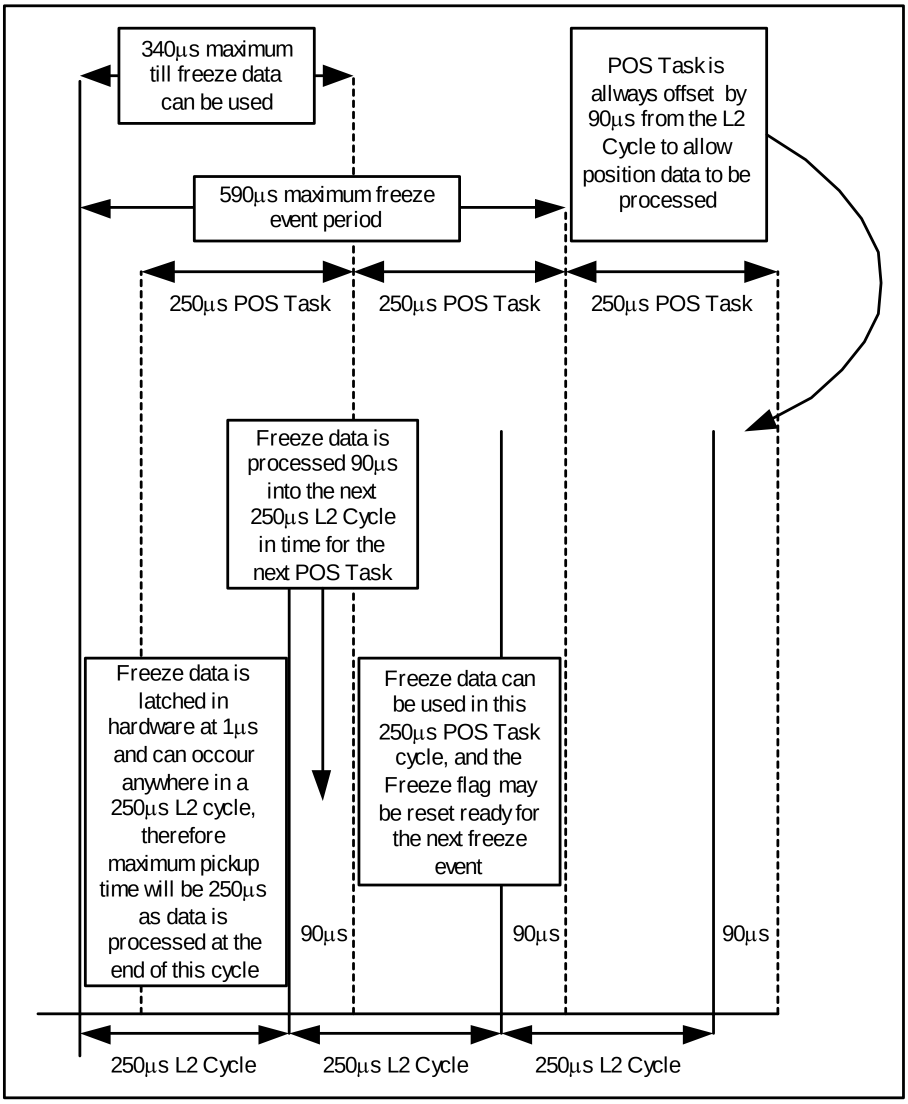
Table 4-5

Configuration	Parameters to set
SM-Applications + SM-Encoder Plus	In this configuration, the Freeze signal is input via the SM-Applications module, on Digital Input 0 (Pin 10). Parameter 81.43 or x.43 sets the detection type, 0 = rising edge, 1 = falling edge. Parameter 81.42 or x.42 must be set to 1 to enable the freeze hardware.
SM-Applications + SM-Universal Encoder Plus	In this configuration, the Freeze signal can be input via the SM-Applications module as above, to freeze the drive encoder position only. The freeze signal can also be input via the SM-Universal Encoder Plus, as this module can freeze the drive encoder position and its own encoder counter position. The Freeze sensor maybe connected to either PL2 Pin 1 (24V), or PL2 Pin 8 and 9 (RS485). Parameter x.38 sets the Freeze signal type, 1 = 24V, 2 = RS485, 3 = 24V + RS485 (logical OR). Parameter x.41 sets the detection type, 0 = rising edge, 1 = falling edge. Parameter x.40 must be set to 1 to enable the SM-Universal Encoder Plus to freeze the main drive position.

Once a Freeze capture has taken place the freeze flags will be set, and will remain latched on, so that further Freeze signals will not be detected until the user resets them. To reset the Freeze flags, use the commands *APCResetRefSourceFreezeFlag()* and *APCResetFbckSourceFreezeFlag()*. These commands reset the all of the freeze flags, including the drive and option module flags, the virtual parameter flags in menu 90, and the APC flags.

The Freeze position data update timing is shown in Figure 4-8 below:

Figure 4-8



4.3.8.3 Marker Position Capture

To action a Marker position capture, the following commands / virtual parameters in Table 4-6 below have to be set:

Table 4-6

Parameter / Command	Function
APCEnableReferenceMarker()	Reference Marker Flag Enable
APCEnableFeedbackMarker()	Feedback Marker Flag Enable
APCResetRefSourceMarkerFlag()	Resets Reference Marker flag
APCResetFbckSourceMarkerFlag()	Resets Feedback Marker flag
APCDisableRefSourceMarker()	Enables Reference Marker Position Capture
APCDisableFbckSourceMarker()	Enables Feedback Marker Position Capture

Like a Freeze position capture, after a Marker capture, the marker flags are set, and must be reset by the commands *APCResetRefSourceMarkerFlag()* and *APCResetFbckSourceMarkerFlag()*, before a new position may be captured.

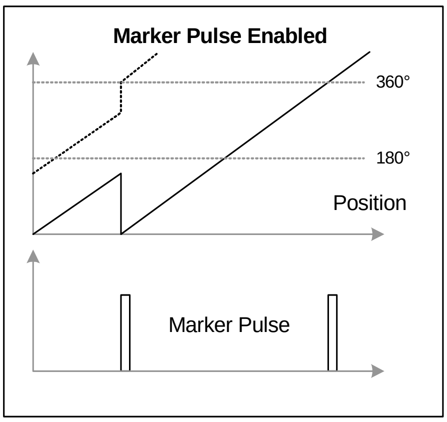
Marker position capture can be used as a second freeze input, provided the correct converters (24V to RS485) are used, and good EMC practices are followed.

If Marker position capture is used, then Marker position reset is not available.

4.3.8.4 Marker Position Reset

When Feedback or Reference, Marker position reset is enabled, the relevant source marker pulse will reset the respective source position counters within one revolution, so that zero position and the marker pulse are aligned, as shown in Figure 4-9 below:

Figure 4-9



If the motor has to turn less than 180° before the marker is found, then the position counters will jump down to zero i.e. zero position turn 0. If the motor has to turn more than 180° before the marker is found, then the position counters will jump up to zero i.e. zero position, turn 1.The turns count is not reset to zero after a marker reset.

To action a Marker position reset, the following commands / virtual parameters shown in Table 4-7 have to be set:

Table 4-7

Parameter / Command	Function
APCEnableReferenceMarker()	Reference Marker Flag Enable
APCEnableFeedbackMarker()	Feedback Marker Flag Enable
APCResetRefSourceMarkerFlag()	Resets Reference Marker flag
APCResetFbckSourceMarkerFlag()	Resets Feedback Marker flag
APCEnableRefSourceMarker()	Enables Reference Marker Position Reset
APCEnableFbckSourceMarker()	Enables Feedback Marker Position Reset

If Marker position Reset is used, then Marker position capture is not available.

4.3.8.5 Freeze Triggering of APC References

The Freeze function may also be used to set the APC Reference selector to either Digital Lock or CAM, selected by the command *APCSelectActionOnFreeze().* This is useful when an application requires that a process must be synchronised to a particular position on the Master axis. See the *Digital lock selection by Freeze* and the *CAM Selection by Freeze*, sections.

To action a Freeze selection, the following commands / virtual parameters in Table 4-8 below have to be set:

Table 4-8

Parameter / Command	Value	Function
APCEnableReferenceFreeze()	NA	Reference Freeze Enable
APCResetRefSourceFreezeFlag()	NA	Resets Reference Freeze flag
APCSelectActionOnFreeze(Action%)	Action% = 1	Digital Lock Select on Freeze
	Action% = 2	CAM Select on Freeze

When a freeze signal is used to trigger the CAM or Digital Lock reference, there is no lost position, from when the freeze was triggered to when the reference is engaged.

4.3.9 Reference and Feedback Disable

The Reference and Feedback may be disabled so that the position feedback can be modified and re-inserted e.g. by a filter. To insert a function block:

1. The Reference or Feedback must be disabled by the commands APCDisableRefInput() and APCDisableFbckInput() respectively.
2. The current source position must be acquired from read parameter [20] or [40] using the APCReadPar(Parameter%) command.
3. The current source position is then fed into the function block, then the new derived position is inserted back into the Reference or Feedback counter using the command APCSetRefInput(Input%) or APCSetFbckInput(Input%) respectively.

If the commands APCEnableRefInput() or APCEnableFbckInput() are active then any value written using APCSetRefInput(Input%) or APCSetFbckInput(Input%) will be ignored, and the position counters will function as normal.

Any values written in POS0 using the commands APCSetRefInput(Input%) or APCSetFbckInput(Input%) are used in the same POS task cycle.

4.4 Stop Reference

The Stop reference is used to stop the Feedback axis, regardless of the previous reference, and current velocity.

The Stop reference has two different operating modes:

4.4.1 Profile Stop

In this mode, *APCSetStopMode(0)*, when the Stop reference is selected, the axis will be ramped from the current speed to zero. This is achieved by the stop reference being loaded with the current position plus the stopping distance. The stopping distance is calculated from the current velocity, and the deceleration rate set in the profile generator, by the command *APCSetProfileDecelRate()*. The Final calculated stopping position is shown by read parameter [91].

4.4.2 Instant Stop

In this mode, *APCSetStopMode(1)*, when the Stop reference is selected, the axis will be decelerated from the current speed to zero instantaneously with no ramps. This is achieved by setting the stop reference to the current position, and bypassing the profile generator. Care should be taken when using this stop mode, that the application mechanics can withstand an instant stop without damage.

4.5 Position Reference

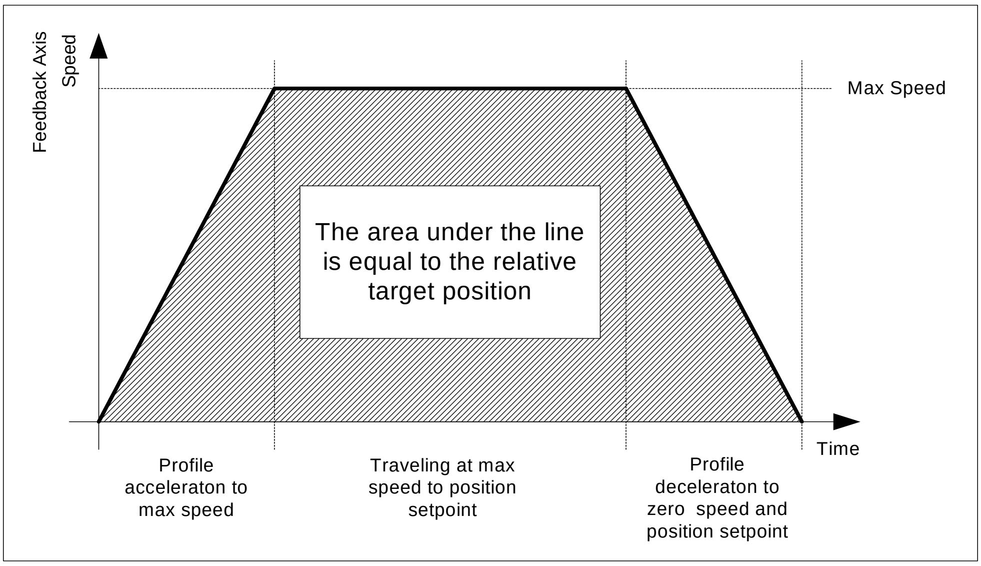
The Position Reference is for simple profile controlled, point to point, indexing moves. The position setpoint may be set by *APCSetPositionSetPoint*, with respect to the current resolution set by *APCSetNumOfTurnsBits*. The position setpoint can be set dynamically while a profile is already in progress e.g. If the position set point is set to 100000, and the feedback axis has travelled to 50000, then if the setpoint is changed to 200000, the Feedback axis will stop at 200000.

The position loop reference will be profiled within the limits of the profile generator, (accel/decel & max profile speed), for the given position setpoint.

Read parameter [92] may be used to view the current accepted position reference.

See Figure 4-10 below:

Figure 4-10



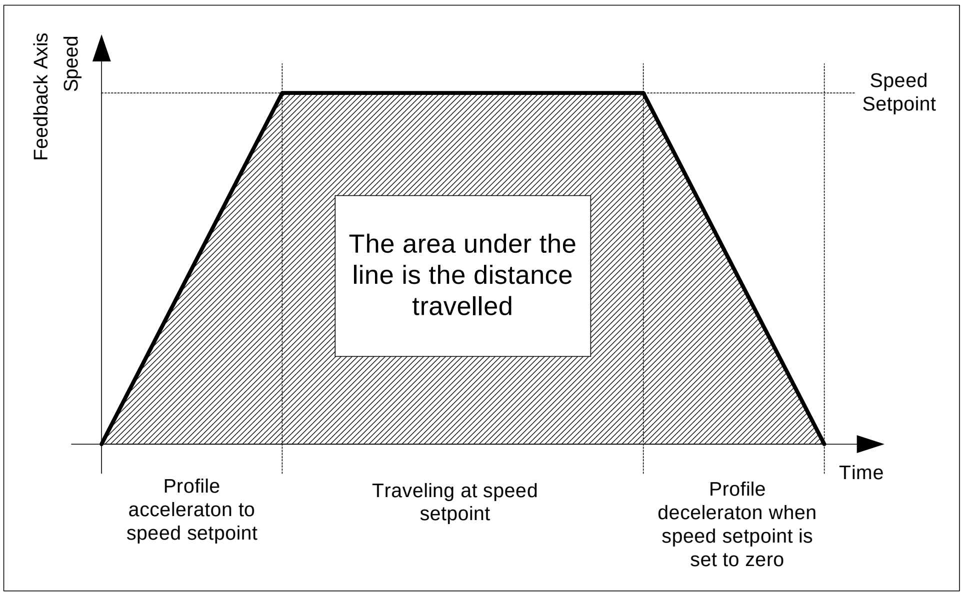
NOTE: If the Position reference is used for in an Indexing application where position change = Current position + Index, the position reference will roll-over with the feedback once -2^{31} to $2^{31}-1$ is exceeded. This means that the user can Index in one direction infinitely, without any unexpected position changes.

4.6 Speed Reference

The Speed reference may be used for jogging and homing type applications. The Speed setpoint is set by *APCSetSpeedSetPoint*, in 2^{32} counts / rev encoder over 250µs.

The position loop reference will be profiled within the limits of the profile generator, (accel/decel & max profile speed), for the given speed setpoint. See Figure 4-11 below:

Figure 4-11

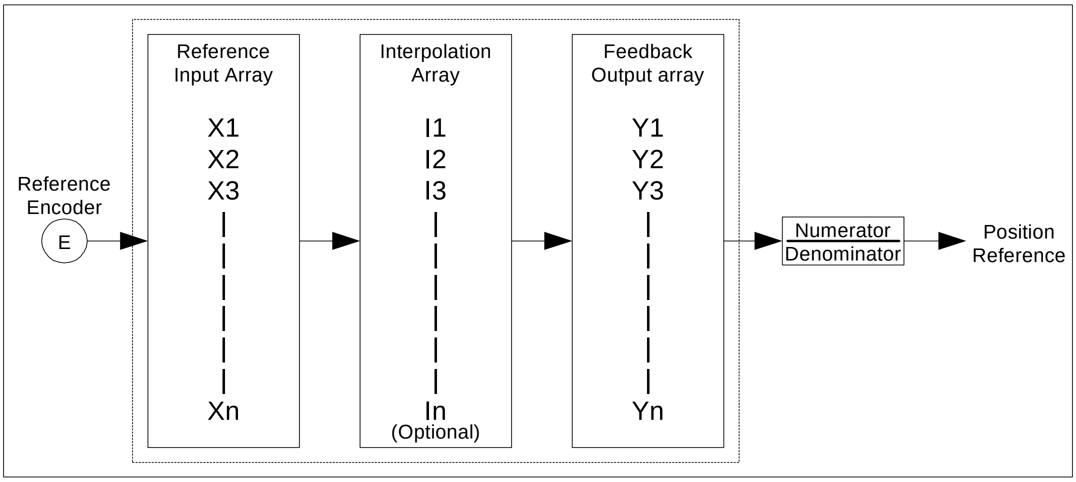


4.7 CAM Reference

4.7.1 Introduction

A mechanical cam produces linear or varied motion from a continuous circular motion i.e. one motion with respect to another. The APC CAM reference works in similar way; a table of Feedback or slave position movements, are calculated with respect to Reference or master position movements, forming a CAM profile table, such that when the CAM reference is selected, the APC produces a varied Feedback axis position reference with respect to Reference axis encoder position, with the given interpolation type selected. See Figure 4-12 below:

Figure 4-12



4.7.2 CAM Co-ordinates

4.7.2.1 Basics

The moves generated by the CAM reference are created from a series of CAM position co-ordinates. These co-ordinates are arranged into Input (Reference) and Output (Feedback) arrays. Arrays are sets of variables of the same type, arranged in order, each variable forming an Array Element, each with a numerical reference. A typical CAM array in DPL might look like:

```
InArray%[0]     = 109227
InArray%[1]     = 218454
InArray%[2]     = 54613
OutArray1%[0]   = 109227
OutArray1%[1]   = 436908
OutArray1%[2]   = 54613
```

The co-ordinates are entered in relative position increments, therefore the feedback absolute position when the CAM has run one cycle, is the sum of all of the feedback co-ordinates in the array. See Table 4-9 below:

Table 4-9

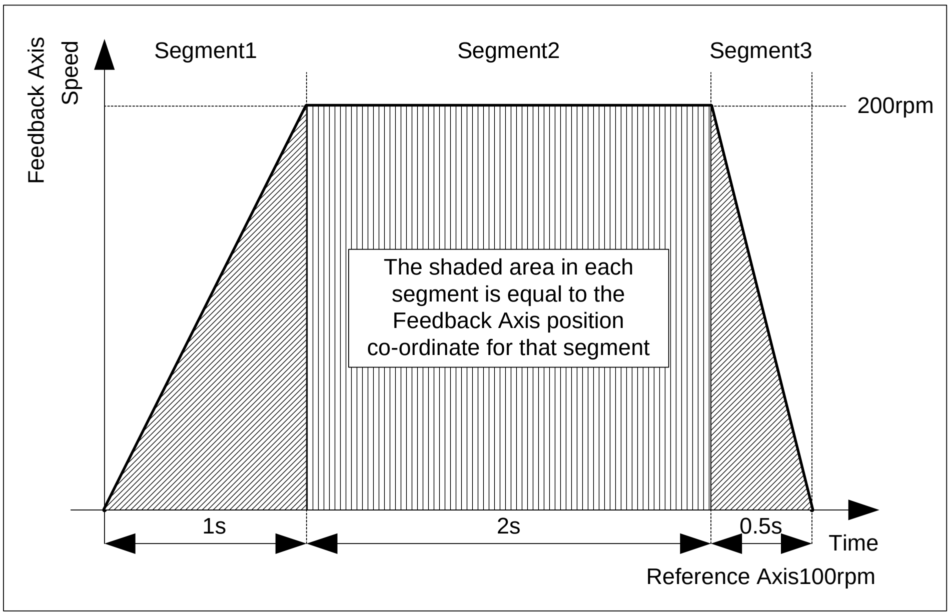
Segment	Relative Position	Absolute Position
1	109227	109227
2	436908	546135
3	54613	600748

The CAM is a Monotonic reference i.e. increments when the reference moves forward, and decrements when the reference moves backward.

4.7.2.2 Calculation Of Co-ordinates

Before the CAM arrays can be calculated, it is necessary to create a speed profile diagram. A typical speed profile diagram is shown in Figure 4-13 below:

Figure 4-13



From the Profile above, we can see that the Feedback axis must accelerate to 200rpm in 1 second, then stay at 200rpm for 2 seconds, and finally decelerate to 0rpm in 0.5 seconds. To determine the amount of position change required for the feedback at the given speed (200rpm), it is necessary to find the integral of each profile segment, or the area under the plot in each segment. The Reference speed from the profile diagram is 100rpm, and the Reference and Feedback encoders will have the same resolution, 65536 counts / rev (number of turns bits = 16). From this it is possible to calculate the Input and output array co-ordinates. First, the number of counts per second at the chosen speed must be found using the following equation:

$$\textbf{Counts / Second} = (\textbf{rpm}/\textbf{60}) \times \textbf{counts / rev}$$

For the Input array:

$$\textbf{Counts / Second} = (\textbf{100}/\textbf{60}) \times \textbf{65536} = \textbf{109227cps}$$

For the Output array:

$$\textbf{Counts / Second} = (\textbf{200}/\textbf{60}) \times \textbf{65536} = \textbf{218454cps}$$

From this we know that the Reference will move by 109227 counts every 1 second, the axis travels at 100rpm, so a 2 second move will be double (218454 counts), and so on. from this we can determine all of the Input array co-ordinates:

```
InArray%[0]     = 109227
InArray%[1]     = 218454
InArray%[2]     = 54613
```

We also know that 200rpm for the Feedback encoder will be 218454 counts/second, so from this we can calculate the Output array co-ordinates using the following equations.

The Acceleration / Deceleration equation is (1/2 v x t, or the area of a triangle: 1/2 base x height):

$$\textbf{Acceleration or Deceleration period} = (\textbf{cps} \times \textbf{Time})/\textbf{2}$$

For the Acceleration period of 1s:

$$\textbf{Acceleration Period} = (\textbf{218454} \times \textbf{1})/\textbf{2} = \textbf{109227}$$

For the Deceleration period of 0.5s

$$\textbf{Deceleration Period} = (\textbf{218454} \times \textbf{0.5})/\textbf{2} = \textbf{54613}$$

The Steady State equation is: (v x t or Area of a square: base x height)

$$\textbf{Steady State} = \textbf{cps} \times \textbf{Time}$$

For the Steady State period of 2s

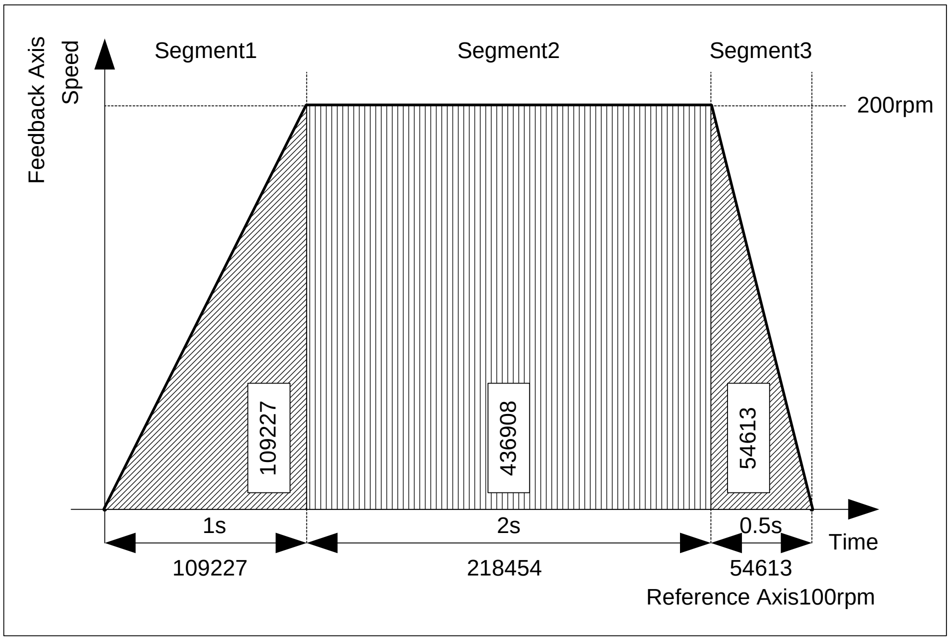
$$\textbf{Steady State} = \textbf{218454} \times \textbf{2} = \textbf{436908}$$

The Output Array co-ordinates are then:

```
OutArray1%[0]   = 109227
OutArray1%[1]   = 436908
Out Array1%[2]  = 54613
```

The completed profile diagram is shown in Figure 4-14 below:

Figure 4-14



For this particular speed profile, Square1 and Square2 interpolation must be used for acceleration and deceleration, and linear used for the steady state section. The interpolation array for the profile would look like this:

```
InterpolationArray%[0] = 2
InterpolationArray%[1] = 0
InterpolationArray%[2] = 3
```

For more information on interpolation, see the *Interpolation* section

If a continuous speed offset is required through the profile, then there is a second output array which is a dedicated output offset array (`OutArray2%`). See Figure 4-15 below:

Figure 4-15

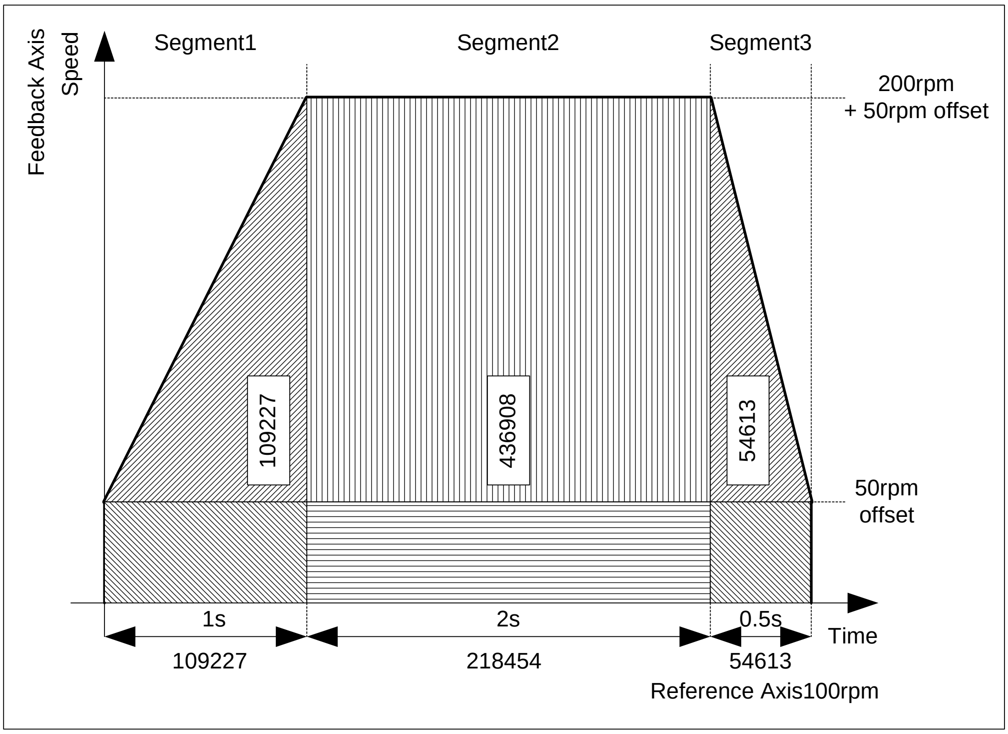


Figure 4-15 above shows the previous speed profile with a 50rpm speed offset. The position co-ordinates for the output offset array is calculated in the same way as the as the steady state section of the profile:

For the Output array:

$$\textbf{Counts / Second} = (\mathbf{50}/\mathbf{60}) \times \mathbf{65536} = \mathbf{54613cps}$$

For the Steady State period of 1s (Segment1)

$$\textbf{Steady State} = \mathbf{54613} \times \mathbf{1} = \mathbf{54613}$$

For the Steady State period of 2s (Segment2)

$$\textbf{Steady State} = \mathbf{54613} \times \mathbf{2} = \mathbf{109227}$$

For the Steady State period of 0.5s (Segment3)

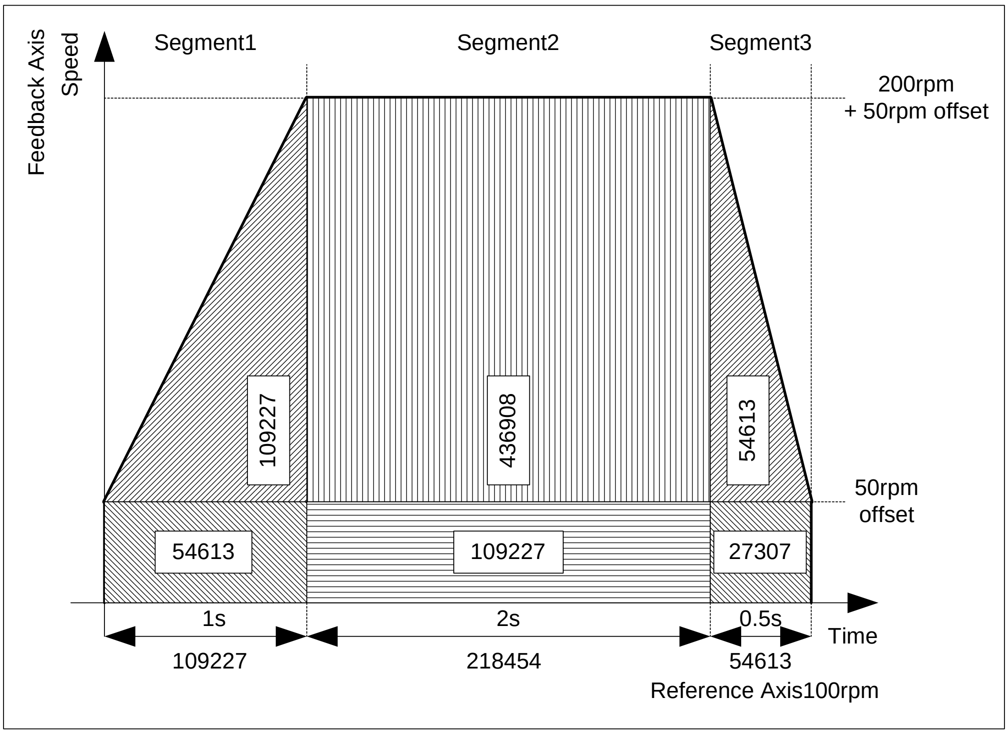
$$\textbf{Steady State} = \mathbf{54613} \times \mathbf{0.5} = \mathbf{27307}$$

The Output Offset Array co-ordinates are then:

```
OutArray2%[0]   = 54613
OutArray2%[1]   = 109227
Out Array2%[2]  = 27307
```

The completed profile diagram is shown in Figure 4-16 below:

Figure 4-16



4.7.3 Interpolation

Interpolation with respect to the CAM reference, is a process of estimating the Feedback position reference within a given CAM segment.

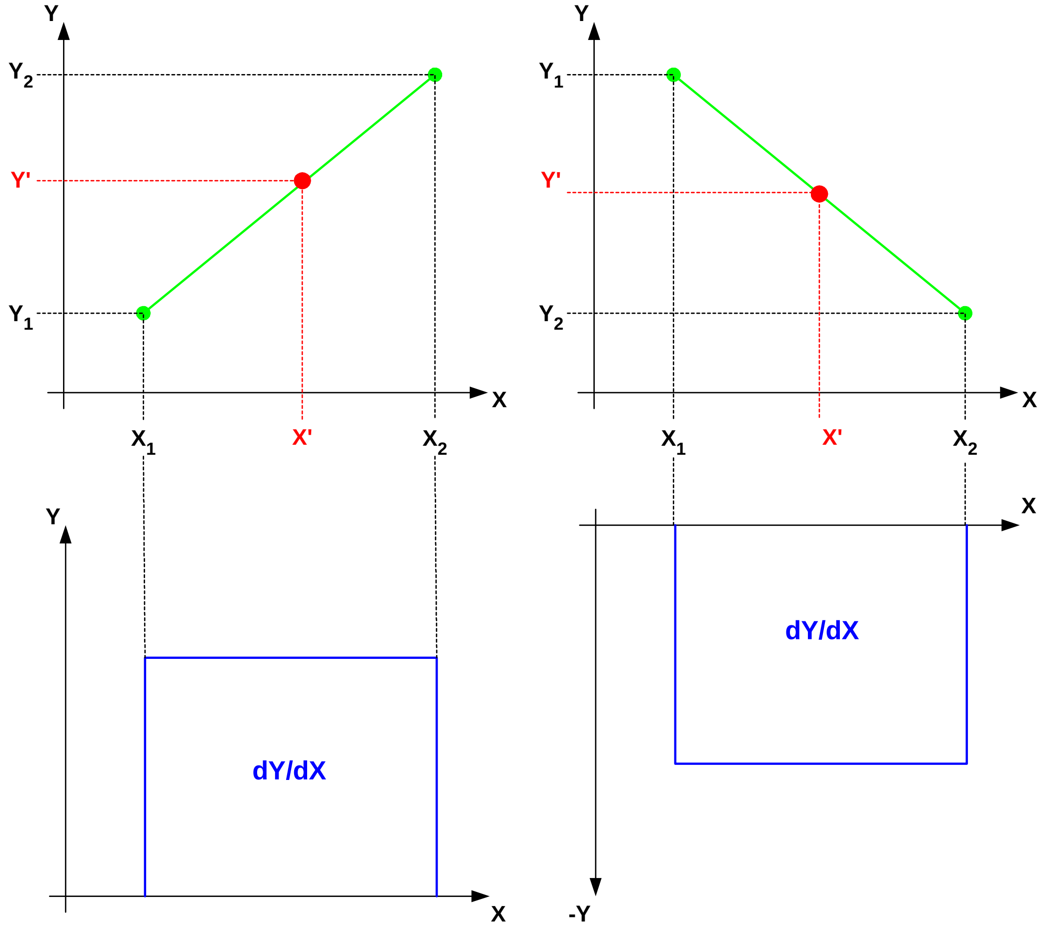
There are 3 different interpolation modes which may be selected for the CAM Reference, using the command *APCSetCAMInterpolationMode(Mode%)*:

- Multiple - *APCSetCAMInterpolationMode(-1)*, where the interpolation used per CAM segment can be selected on an individual basis, defined by the interpolation array.
- Linear - *APCSetCAMInterpolationMode(0)*, where only Linear interpolation will be used in all CAM segments.
- Cosine - *APCSetCAMInterpolationMode(1)*, where only Cosine interpolation will be used in all CAM segments

Multiple interpolation incorporates both Linear and Cosine interpolation, together with Square1, Square2, Cosine1 and Cosine2. All of these interpolation methods are described below:

4.7.3.1 Linear Interpolation (0):

Figure 4-17



When Linear Interpolation has been applied to a CAM segment, it will derive a steady state velocity profile between CAM co-ordinates. It is used for steady state velocity profiles. This means that if linear interpolation is used over the whole CAM there will be a step change in velocity between CAM segments.

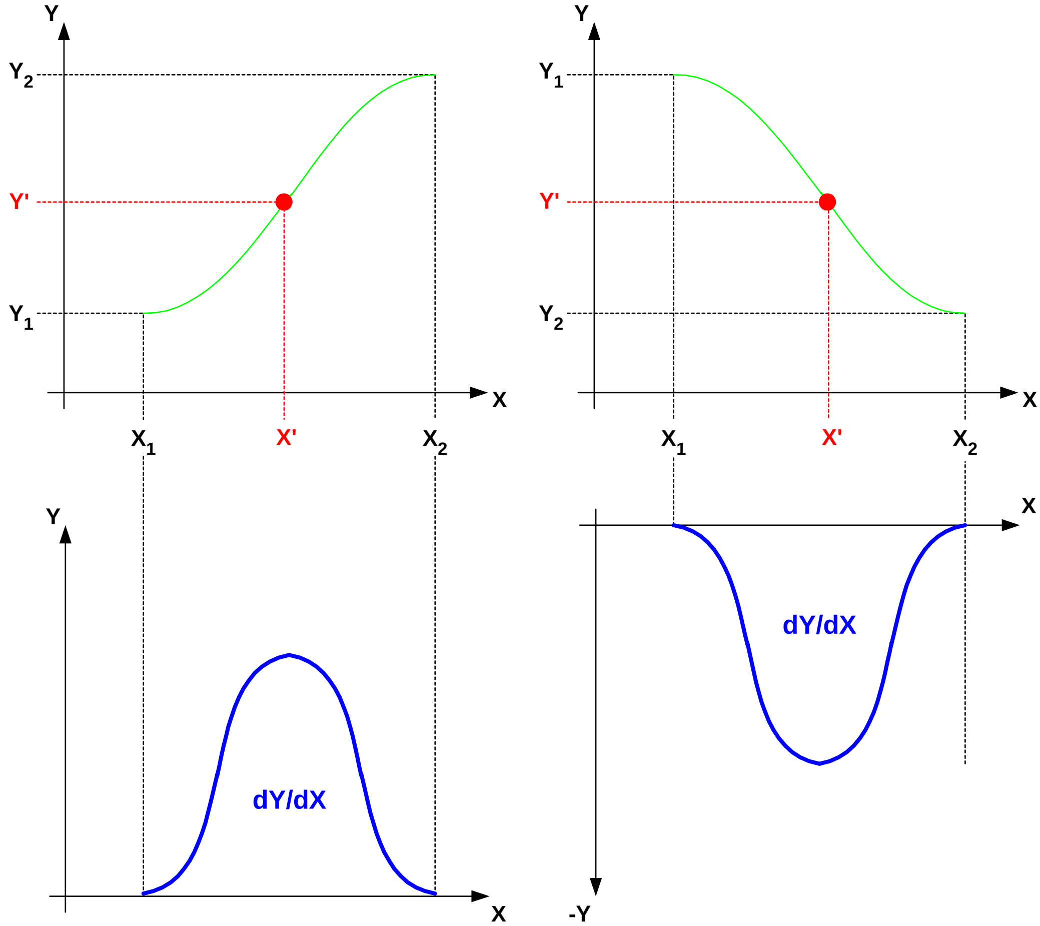
In the Upper half of Figure 4-17 above, Linear Interpolation has been used between each set of co-ordinates (X,Y), to determine the value of Y' position, with respect to X'. The Lower half of Figure 4-17 above is the rate of change of linear interpolation, and indicates the resultant velocity, when Linear Interpolation is used.

The final Slave axis velocity is equal to:

Y (velocity) = X (velocity) * ((Y co-ordinate)/(X co-ordinate))

4.7.3.2 Cosine Interpolation (1):

Figure 4-18



When Cosine Interpolation has been applied to a CAM segment, it will derive a sinusoidal velocity profile between CAM co-ordinates. It is used for smooth point to point position changes.

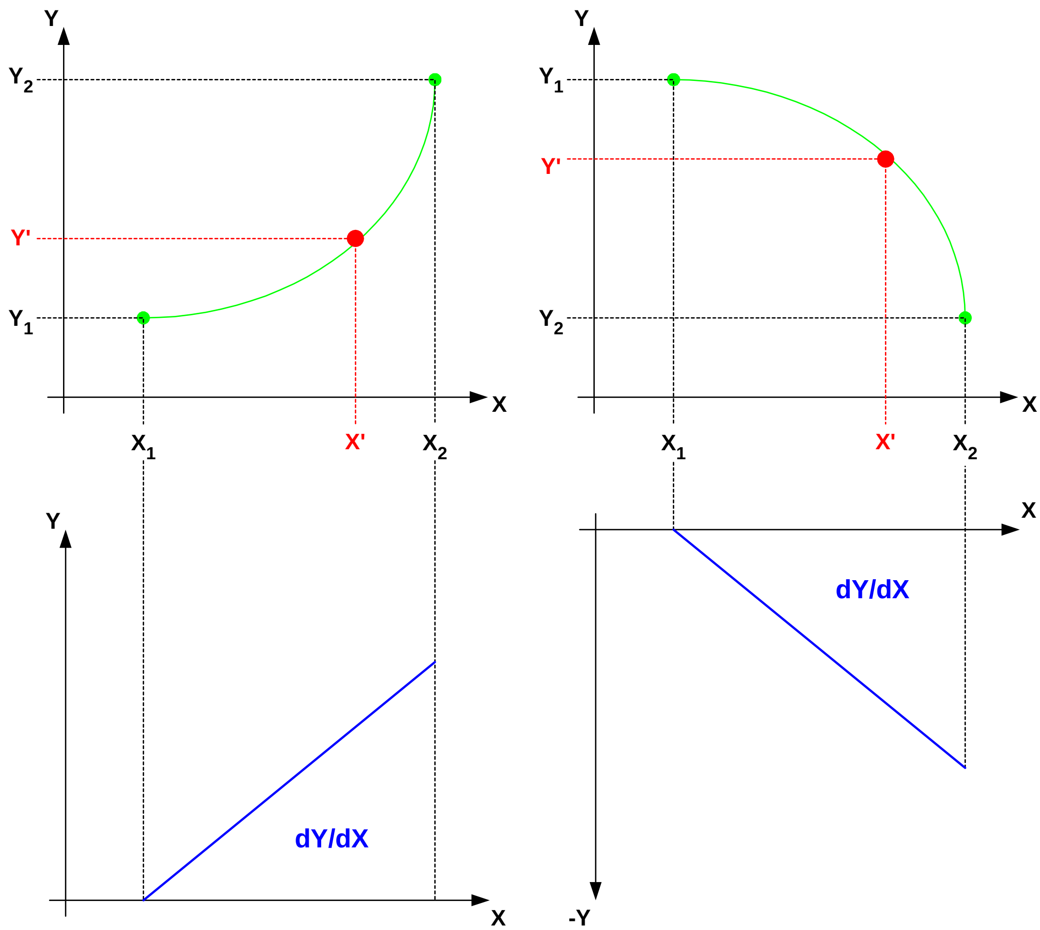
In the Upper half of Figure 4-18 above, Cosine Interpolation has been used between each set of co-ordinates (X,Y), to determine the value of Y' position, with respect to X'. The Lower half of Figure 4-18 above is the rate of change of Cosine interpolation, and indicates the resultant velocity, when Cosine Interpolation is used.

The final Slave axis velocity is equal to:

Y (velocity) = X (velocity) * ((Y co-ordinate)/(2*X co-ordinate))

4.7.3.3 Square1 Interpolation (2):

Figure 4-19



When Square1 Interpolation has been applied to a CAM segment, and the X axis is positive and increasing, it will derive a positive, linear acceleration, velocity profile. It is used for linear velocity acceleration over a given CAM segment.

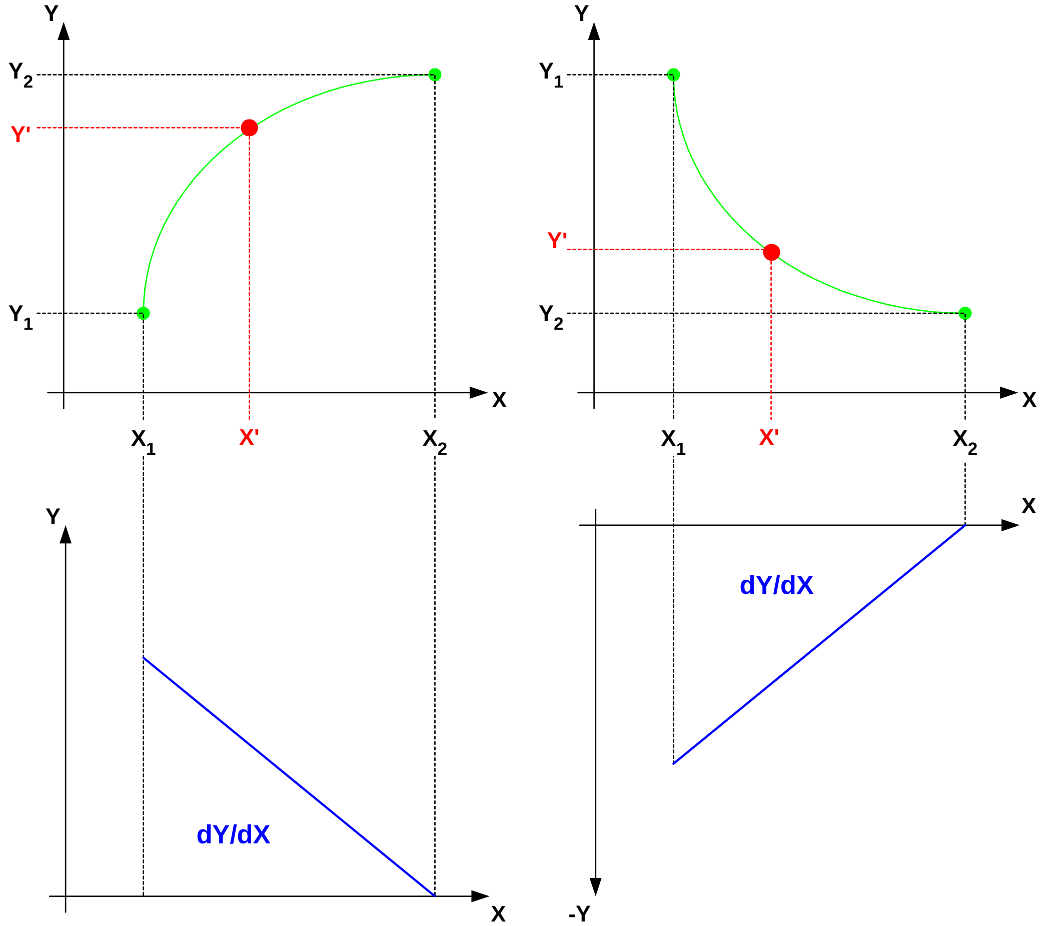
In the Upper half of Figure 4-19 above, Square1 Interpolation has been used between each set of co-ordinates (X,Y), to determine the value of Y' position, with respect to X'. The Lower half of Figure 4-19 above is the rate of change of Square1 interpolation, and indicates the resultant velocity, when Square1 Interpolation is used.

The final Slave axis velocity is equal to:

Y (velocity) = X (velocity) * ((2*Y co-ordinate)/(X co-ordinate))

4.7.3.4 Square2 Interpolation (3):

Figure 4-20



When Square2 Interpolation has been applied to a CAM segment, and the X axis is positive and increasing, it will derive a negative, linear acceleration, velocity profile.It is used for linear velocity deceleration over a given CAM segment.

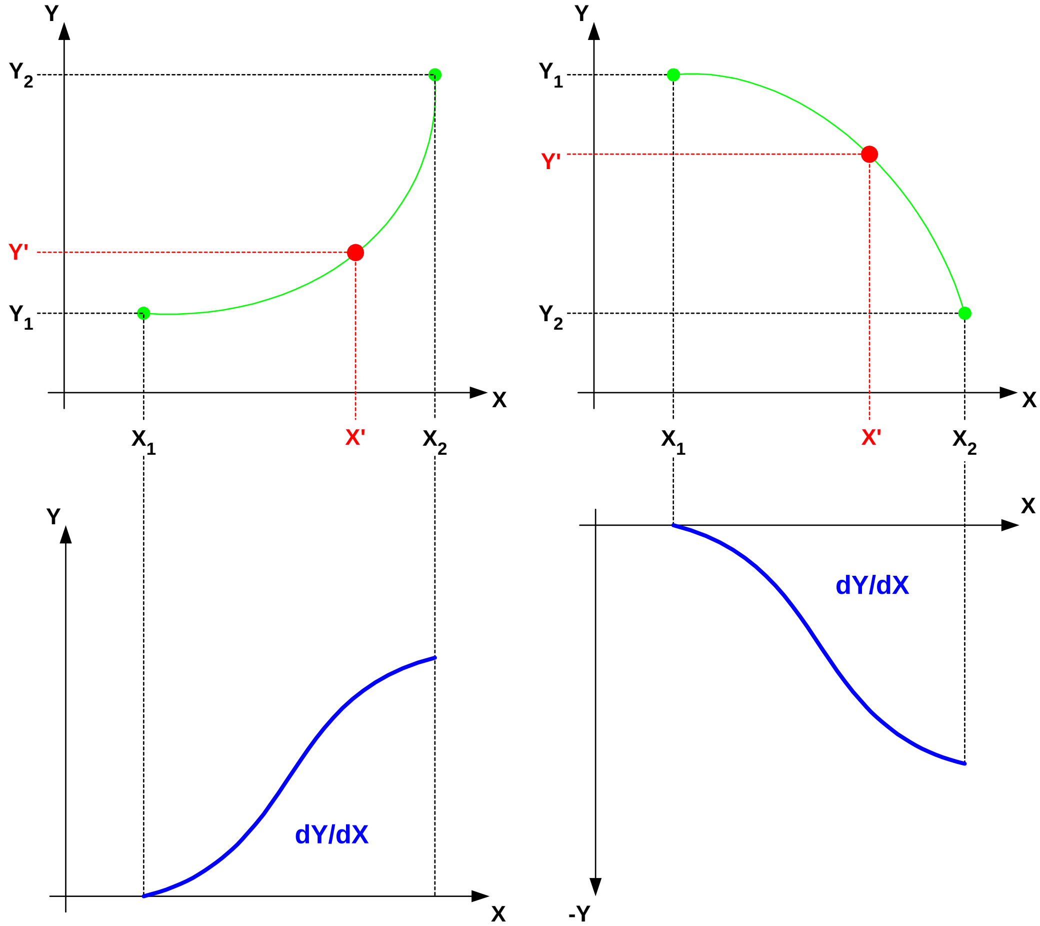
In the Upper half of Figure 4-20 above, Square2 Interpolation has been used between each set of co-ordinates (X,Y), to determine the value of Y' position, with respect to X'. The Lower half of Figure 4-20 above is the rate of change of Square2 interpolation, and indicates the resultant velocity, when Square2 Interpolation is used.

The final Slave axis velocity is equal to:

Y (velocity) = X (velocity) * ((2*Y co-ordinate)/(X co-ordinate))

4.7.3.5 Cosine1 Interpolation (4):

Figure 4-21



When Cosine1 Interpolation has been applied to a CAM segment, and the X axis is positive and increasing, it will derive a positive, sinus acceleration, velocity profile. It is used for sinusoidal velocity acceleration over a given CAM segment.

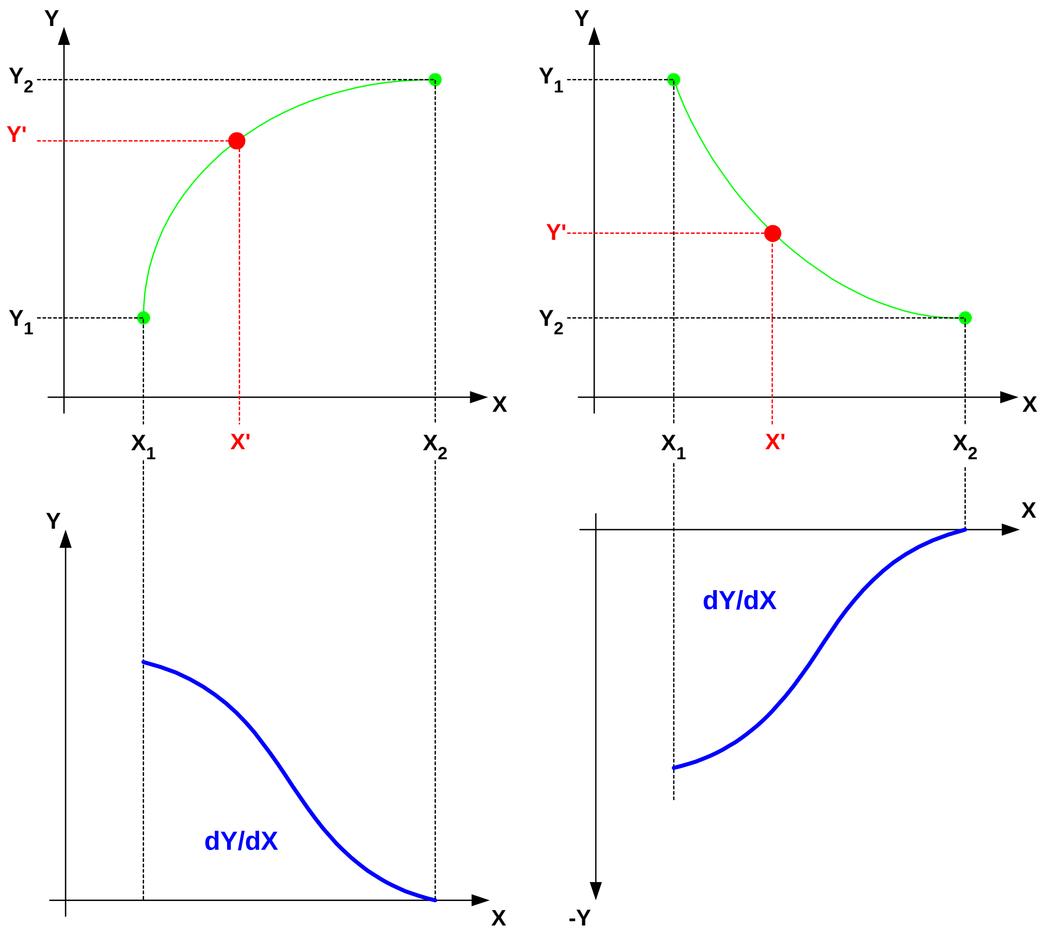
In the Upper half of Figure 4-21 above, Cosine1 Interpolation has been used between each set of co-ordinates (X,Y), to determine the value of Y' position, with respect to X'. The Lower half of Figure 4-21 above is the rate of change of Cosine1 interpolation, and indicates the resultant velocity, when Cosine1 Interpolation is used.

The final Slave axis velocity is equal to:

Y (velocity) = X (velocity) * ((2*Y co-ordinate)/(X co-ordinate))

4.7.3.6 Cosine2 Interpolation (5):

Figure 4-22



When Cosine2 Interpolation has been applied to a CAM segment, and the X axis is positive and increasing, it will derive a negative, sinus acceleration, velocity profile. It is used for sinusoidal velocity deceleration over a given CAM segment.

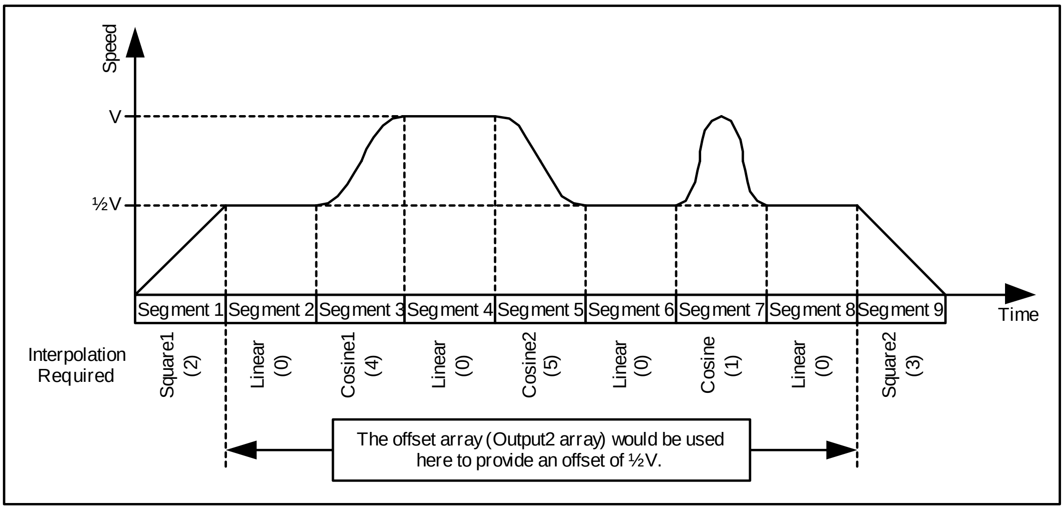
In the Upper half of Figure 4-22 above, Cosine1 Interpolation has been used between each set of co-ordinates (X,Y), to determine the value of Y' position, with respect to X'. The Lower half of Figure 4-22 above is the rate of change of Cosine2 interpolation, and indicates the resultant velocity, when Cosine1 Interpolation is used.

The final Slave axis velocity is equal to:

Y (velocity) = X (velocity) * ((2*Y co-ordinate)/(X co-ordinate))

Figure 4-23 below shows a profile, and the type of interpolation required to perform that profile.

Figure 4-23



Note: For a linear acceleration/deceleration velocity profile, the actual change in position will be a square function, as the integral of the speed over a time period gives the position change, and the integral of a ramp (accelerating or decelerating), is a square function.

4.7.4 Constant and Variable Arrays

The arrays which hold the CAM data may be defined as Constant, or Variable.

Constant arrays should be used when the values in an array do not need to be changed whilst the program code is running. Constant arrays are placed into Flash ROM, and therefore can only be altered by changing the array values in the DPL program code. A constant array is declared by using the CONST command; see the SYPT user help guide, under “Arrays/Constants” for examples on how to enter constant arrays.

Variable arrays should be used when the values in an array need to be changed whilst the program code is running. Variable arrays are placed into RAM. A Variable array has to be dimensioned so that the DPL compiler can reserve space in RAM for the Data. A Variable array is dimensioned using the DIM command; see the SYPT user help guide, under “Arrays” for examples on how to enter Variable arrays.

4.7.5 CAM Initialisation

Once the CAM Co-ordinates have been calculated, and the interpolation type has been chosen the CAM arrays should be Initialised by **one** the following commands:

- *APCCamInitialise* - Initialises the CAM with an Input, Output 1, Output 2, and Interpolation array.
- *APCCamInitialise1* - Initialises the CAM with an Input and Output 1 array.
- *APCCamInitialise2* - Initialises the CAM with an Input, Output 1, and Interpolation array.
- *APCCamInitialise3* - Initialises the CAM with an Input, Output 1, and Output 2 array.

These commands act as a pointer for the CAM reference so that the location of the various arrays is known.

The CAM start index and CAM Size must be set, to define where in the CAM arrays you want to start from, and how much of the CAM, from the start point, you want to use. The CAM start index and CAM Size is set by the commands:

- *APCSetCAMStartIndex*
- *APCSetCAMSize*

The CAM Start Index, and the CAM size, can be changed whilst the CAM reference is running, but will not take effect until the current cycle has finished, or the CAM is re-started.

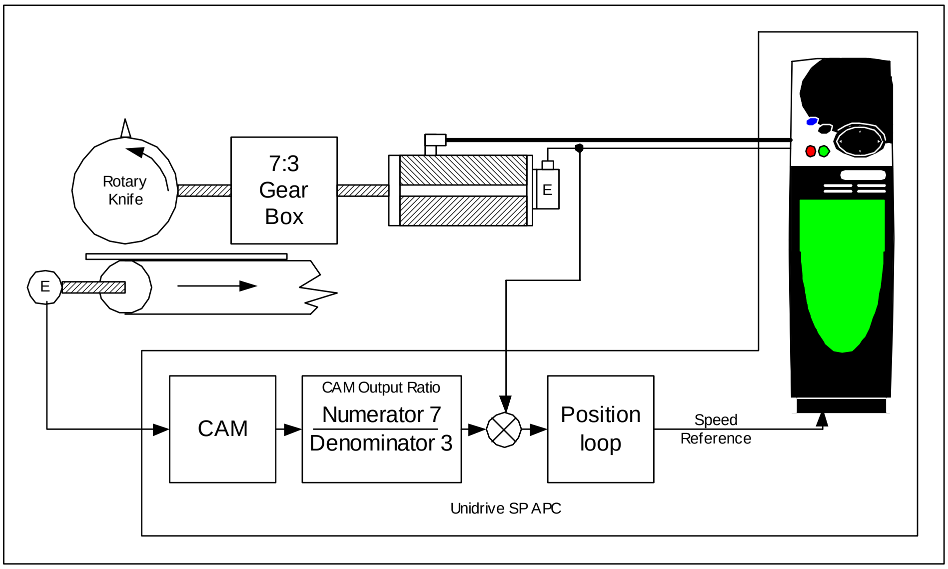
NOTE Any changes to the CAM array data will take effect immediately. Care must be taken to ensure that the array elements being changed are not being used, to prevent sudden unexpected jumps in position, which could damage the application mechanics or personnel.

The CAM Delta segment limit must also be set, to define the maximum change allowed in the Reference source per POS task cycle, set by #81.12 or x.12 for the SM-Applications module. Setting this to a suitable value, will prevent large step changes in reference source, from being applied to the Slave axis, which could potentially damage an applications mechanics. The CAM Delta Segment limit is set by the command APCSetCAMDeltaSegLimit.

4.7.6 CAM Output Ratio

The CAM reference has an output ratio, in form of a Numerator and Denominator. This can be used to compensate for a gear box in a CAM application, like a Rotary Knife application shown in Figure 4-24 below:

Figure 4-24



This means that the CAM can be calculated at 1:1 ratio, therefore minimising any count rounding errors. The gear box ratio is then compensated for by the CAM output ratio.

The CAM output ratio is set by the following commands:

- *APCSetCAMOutRatioNumerator*
- *APCSetCAMOutRatioDenominator*

If granularity of the position output at the motor shaft is a problem when the CAM output numerator is greater than 1 e.g. 7:3, the slave co-ordinates can be multiplied by the numerator value, 7, and the CAM output ratio can be reduced to 1:3 (using the ratio to apply the denominator means that any remainder is automatically compensated for). This means that instead of each count being directly multiplied i.e. a step jump of 7 counts for 1 count in, the CAM will interpolate through the 7 counts, giving a smooth transition with low granularity.

4.7.7 Single Shot and Continuous CAM

The CAM may be run in of two modes:

4.7.7.1 Single Shot

In single shot mode, the CAM will run from the selected start co-ordinate, and will continue until the last co-ordinate is reached, and then stop. The CAM then has to be re-started by one of the following methods:

- Perform an *APCReset*
- Disable single shot mode by using *APCDisableCAMSingleShot*, then re-enable using *APCEnableCAMSingleShot*.
- De-select, then reselect the CAM Reference using *APCSelectReference*
- De-select, then reselect APC run mode 2 using *APCSetRunMode*

Single shot mode may be selected at any point whilst the CAM is running in continuous mode, and will cause the CAM to run until the end of the current cycle.

4.7.7.2 Continuous

In Continuous mode, the CAM will run cyclically, running till the last co-ordinate, then wrapping around to the first co-ordinate again. Continuous CAM mode is selected by *APCDisableCAMSingleShot*. If the start index is changed, it will not take effect until the current CAM cycle has been completed.

4.7.8 Zero and Absolute CAM Reset

It is possible to select within the defined CAM, where to start from, when the CAM reference is next selected:

4.7.8.1 Zero CAM Reset

When the CAM reference is deselected, and Zero CAM reset has been selected by *APCSelectCAMZeroReset*, if the CAM Reference is next selected, the CAM will start from the segment defined by *APCSetCAMStartIndex*.

The CAM Start Index, and the CAM size, *APCSetCAMSize*, can be changed whilst the CAM reference is running, but will not take effect until the current cycle has finished, or the CAM is re-started.

4.7.8.2 Absolute CAM Reset

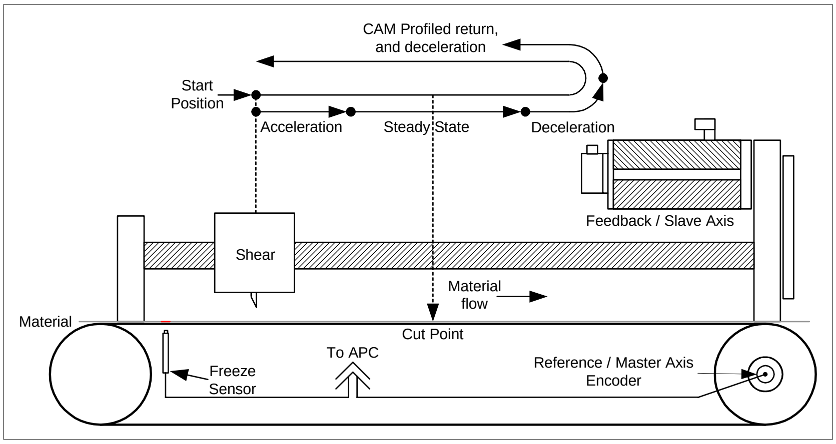
When the CAM reference is deselected, and Absolute CAM reset has been selected by *APCSelectCAMAbsoluteReset*, if the CAM Reference is next selected, the CAM will start from the segment, and the position in the segment, defined by *APCSetCAMAbsResetIndex*, and *APCSetCAMAbsResetPositionInSeg*, within the limits defined by *APCSetCAMStartIndex*, and *APCSetCAMSize.*

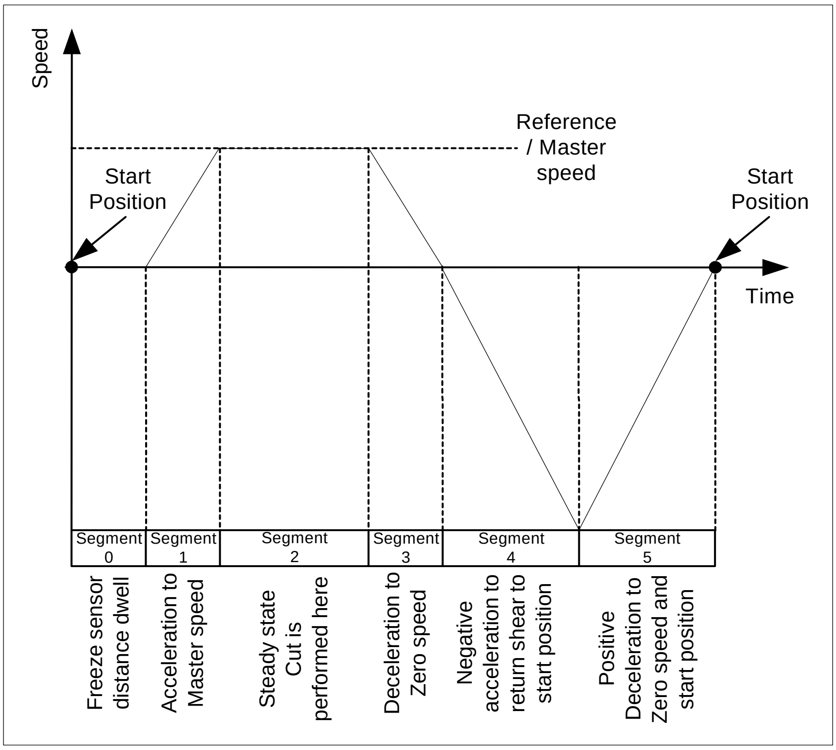
This may be useful for absolute positioning when it is important to ensure the Feedback axis / CAM is set to the correct position after an unforeseen power cut, or reset.

4.7.9 CAM Selection by Freeze

The CAM reference may be selected by a freeze input. This is useful when an application requires that a process must be synchronised to a particular position on the Reference / Master axis, like a flying shear application, shown in Figure 4-25:

Figure 4-25





In this type of flying shear the distance from the freeze sensor, to the shear cutting blade, and the acceleration to the master speed, must be compensated for when calculating the profile. The advantage of this type of flying shear, unlike a Digital Lock flying shear:

- No time is wasted when synchronising the Slave to the Master axis.
- There is no over speed required to recover lost acceleration position.
- The slave is permanently locked in position with the master at all times.

4.7.10 CAM Single Shot Freeze Re-Arm

This CAM single shot freeze re-arm feature is for basic registration systems implemented with the CAM reference in single shot, where a registration sensor is used to trigger the reference freeze input and thereby start the CAM, like the one shown in section 4.7.9 *CAM Selection by Freeze* . When enabled, and a single shot completes, the stop reference will automatically be selected, the reference freeze will be reset, and the freeze trigger system will be reset, ready for a freeze signal to trigger the CAM again. This feature saves the user 1 Pos task cycle by internally resetting the system, saving the user from.

To use this feature the user must:

- Select single shot CAM with APCEnableCAMSingleShot(), before starting the CAM on a freeze for the first cycle, every time the system is run.
- Enable the reference freeze with APCEnableReferenceFreeze(), in the initial task.
- Reset the Freeze flag with APCResetRefSourceFreezeFlag() before starting the CAM on a freeze for the first cycle, every time the system is run.
- Select action on freeze with APCSelectActionOnFreeze(2) before starting the CAM on a freeze for the first cycle, every time the system is run.

This feature is intended to be used on systems where the distance between cuts or cut length etc., is always greater than the total master distance used through the whole CAM profile. If for example the cut length is shorter than the total master distance, then cuts will be missed.

4.7.11 Calculating CAM Co-ordinates from Slave Positional Information

This method demonstrates how to calculate the Master co-ordinates for a trapezoidal Velocity profile from the following position information: -

1. Master Overall Distance (Pmt)
2. Slave Overall Distance (Pst)
3. Slave Acceleration Distance (Psa)
4. Slave Deceleration Distance (PSG)

With reference to Figure 4-26 below the following calculation can be deduced: -

Slave Steady State Distance (Pss) = Pst - Psa - Psd

The Total Equivalent Slave area (Psf) = (2*Psa) + Pss + (2*Psd)

Master Acceleration Distance (Pma) = (2 * Psa / Psf) * Pmt

Master Steady State Distance (Pms) = (Pss / Psf) * Pmt

Master Deceleration Distance (Pmd) = (2 * Psd / Psf) * Pmt

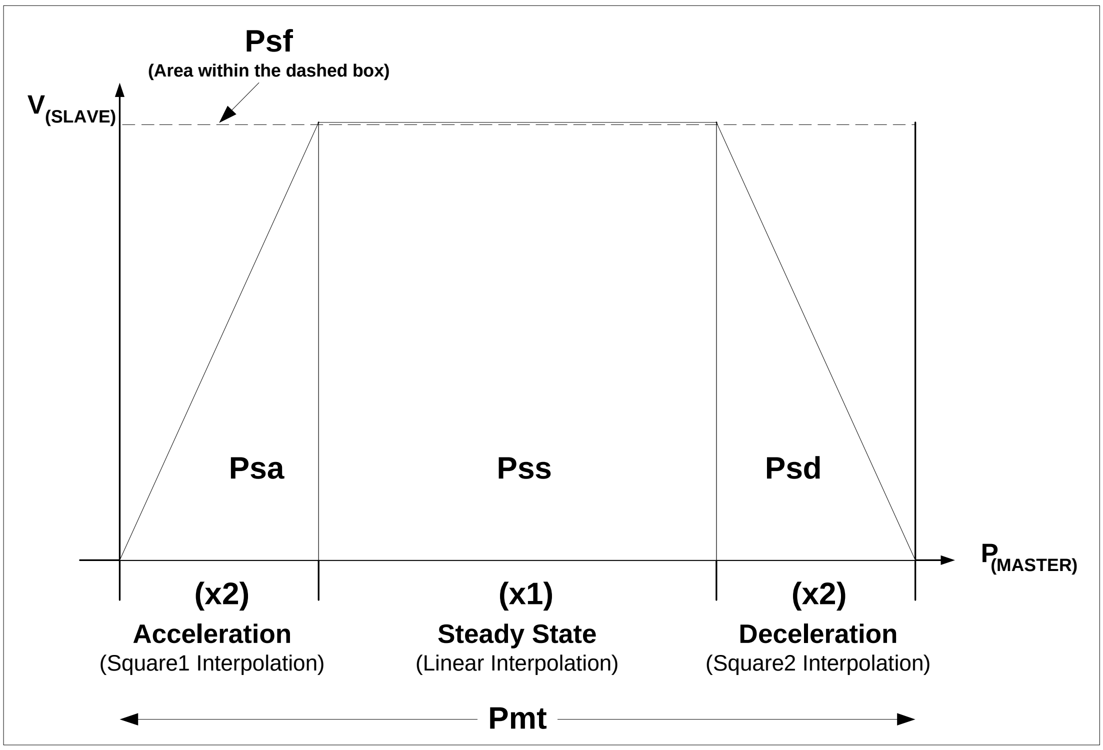
To ensure the master distances calculated are correctly, the master velocities can be checked i.e. If the master distances have been calculated correctly, all 3 equations below will have the same answer:

Final velocity for acceleration = Master Velocity * ((2 * Psa) / Pma)

Steady State Velocity = Master Velocity * (Pss / Pms)

Start velocity for deceleration = Master Velocity * ((2 * Psd) / Pmd)

Figure 4-26



4.7.12 Calculating CAM Co-ordinates from time and distance Information

This method demonstrates how to calculate the Master and Slave co-ordinates for a trapezoidal Velocity profile from the following position information: -

Acceleration time (A1)
Constant speed time (A2)
Deceleration time (A3)
Slave total distance (Pst)
Velocity of master (Vm)

With reference to Figure 4-26 the following calculation can be deduced:

Total Acceleration time (At) = (A1/2) + A2 + (A3/2)
Slave Acceleration Distance (Psa) = (((A1 / At) * Pst) / 2)
Slave Steady State Distance (Pss) = ((A2 / At) * Pst)
Slave Deceleration Distance (Psd) = (((A3 / At) * Pst) / 2)

Amt = A1 + A2 + A3

Total distance of the master will be:

Pmt = Amt * Vm

Master Acceleration Distance (Pma) = ((A1 / Amt) * Pmt)
Master Steady State Distance (Pms) = ((A2 / Amt) * Pmt)
Master Deceleration Distance (Pmd) = ((A3 / Amt) * Pmt)

The Constant speed slave Velocity will be

Vs = Pss / A2

Check that your calculations are correct:

Pst = Psa + Pss + Psd
Pmt = Pma + Pms + Pmd

4.8 Digital Lock Reference

4.8.1 Introduction

Digital Lock is a common industrial method of synchronising a Slave (feedback) axis to a Master (reference) axis. There are two different ways in which the Digital Lock reference can be used:

- “Gear Box” lock - where the Slave is effectively directly coupled to the Master motor, and both motors start from zero speed.
- Master Synchronisation - where the Master motor is at some velocity, and the Slave must synchronise (Lock) with the master in a controlled fashion using ramps, with or without position recovery for acceleration (Rigid or Non Rigid Lock respectively).

4.8.2 Digital Lock Mode

There are three different digital lock modes, Unlocked, Locked, and Never Lock.

4.8.2.1 Unlocked:

In Unlocked mode the profile parameters (Acceleration rate, Deceleration rate, and Max Speed), are used to obtain a lock to the master/reference axis. Once the locking conditions are set (by Rigid or Non Rigid digital lock), the mode will automatically switch to Locked. When the Digital Lock reference is selected, using the reference selector, or by a Freeze input, the default mode is Unlocked.

Command References:

APCSetProfileAccelRate

APCSetProfileDecelRate

APCSetProfileMaxSpeedClamp

APCEnableRigidLock

APCDisableRigidLock

APCSetDigLockMode

APCSelectReference

APCSelectActionOnFreeze

4.8.2.2 Locked:

In Locked mode the profile generator is bypassed, so that any changes in the reference axis speed and position will be instantly responded to by the APC. It is possible to manually switch into Locked mode, however care should be taken that the axes are synchronised when this happens to avoid large instantaneous jumps in position, which may damage application mechanics.

Command References:

APCSetDigLockMode

4.8.2.3 Never Lock:

In Never Lock mode, like Unlocked mode the profile parameters (Acceleration rate, Deceleration rate, and Max Speed), are used to synchronise the reference and feedback axes, however once the axes have synchronised the profile generator will remain active, therefore any large changes in Reference position will be smoothed out by the profile generator on the Feedback axis. Never lock mode is only accessed by manually setting the Digital Lock mode to Never Lock.

Command References:

APCSetProfileAccelRate

APCSetProfileDecelRate

APCSetProfileMaxSpeedClamp

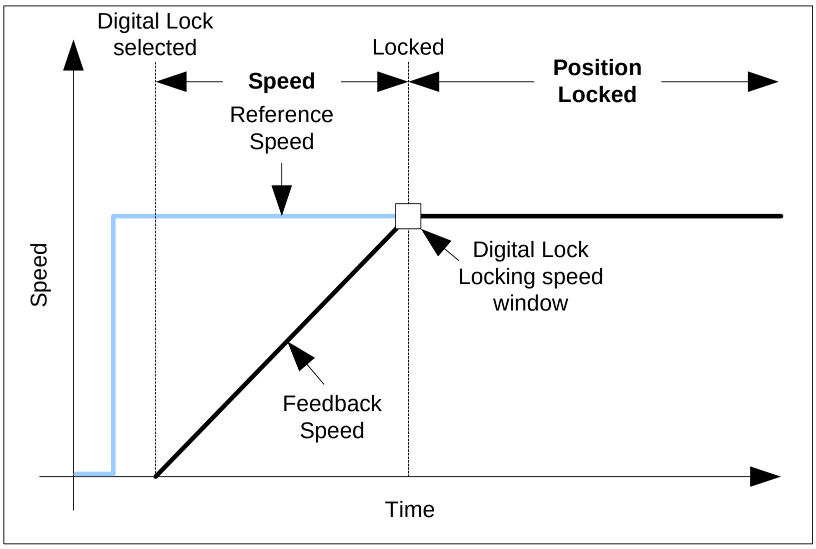
APCSetDigLockMode

4.8.3 Non Rigid Digital lock

When Non Rigid Digital Lock is active, and there is a difference in speed between the Master and Slave velocity; when the Slave is enabled the slave will ramp up to the Master speed, and become locked to the master axis position when the Digital lock, locking speed window is reached. No position will be recovered from when the Slave accelerated / decelerated to the Master speed. If the Slave is enabled when the Master is at zero speed, then the Slave will become locked directly.

To attain a Locked status from an Unlocked status, whilst in Non Rigid Digital Lock, the profile generator must provide a profile that will give the reference speed only, within the limit set by the Maximum profile speed, Profile acceleration /deceleration, and Digital Lock Locking Speed. See Figure 4-27below:

Figure 4-27



If Never Lock mode has been selected, the slave will catch up to the master speed, but the Digital lock mode will not change to locked, and the Profile Generator will remain active and any abrupt changes in Master speed will be ramped to by the Slave i.e. the Reference will be followed within the bounds of the profile acceleration / deceleration rate, and the profile maximum speed clamp.

Command References:

APCDisableRigidLock

APCSetDigLockLockingSpeed

APCSetDigLockMode

APCSetProfileAccelRate

APCSetProfileDecelRate

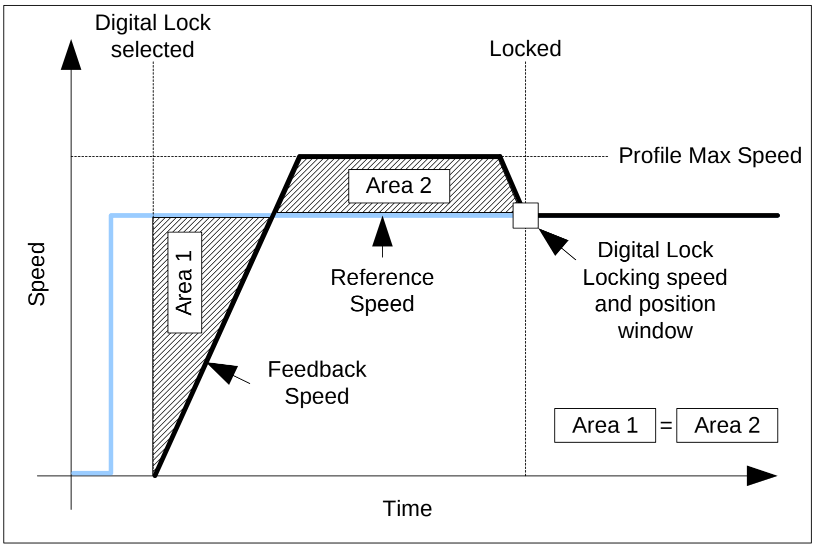
APCSetProfileMaxSpeedClamp

4.8.4 Rigid Digital Lock

When Rigid Digital Lock is active, and there is a difference in speed between the Master and Slave velocity; when the Slave is enabled the slave will ramp up to the maximum profile speed, to regain position lost during acceleration, then ramp down to the Master speed, and become locked to the master axis position when the Digital lock, locking speed and position window is reached. If the Slave is enabled when the Master is at zero speed, then the Slave will become locked directly.

To attain a Locked status from an Unlocked status, whilst in Rigid Digital Lock, the profile generator must provide a profile that will give the reference speed and position, within the limits set by the Maximum profile speed, Profile acceleration /deceleration, Digital Lock Locking Speed and Position. See Figure 4-28 below:

Figure 4-28



If Never Lock mode has been selected, the slave will catch up to the master speed and relative position, but the Digital lock mode will not change to locked, and the Profile Generator will remain active, and any abrupt changes in Master speed will be ramped to by the Slave i.e. the Reference will be followed within the bounds profile acceleration / deceleration rate, and the profile maximum speed clamp.

NOTE If a non-rigid digital lock synchronisation is in progress, and rigid digital lock is selected, the slave will synchronise to the master speed and the relative position at which rigid digital lock was selected.

Command References:

APCEnableRigidLock

APCSetDigLockLockingSpeed

APCSetDigLockLockingPosition

APCSetDigLockMode

APCSetProfileAccelRate

APCSetProfileDecelRate

APCSetProfileMaxSpeedClamp

4.8.5 Digital Lock Ratio

The Digital lock reference has an adjustable output ratio in the form of a Numerator and Denominator, to allow the Digital lock reference to be used as a “Virtual Gear box”, or to compensate for mechanical gearing in a system.

Command References:

APCSetDigLockRatioNumerator

APCSetDigLockRatioDenominator

When using numerator and denominator ratio entry to simulate or replace a mechanical gearbox, there are two types of data which may be entered:

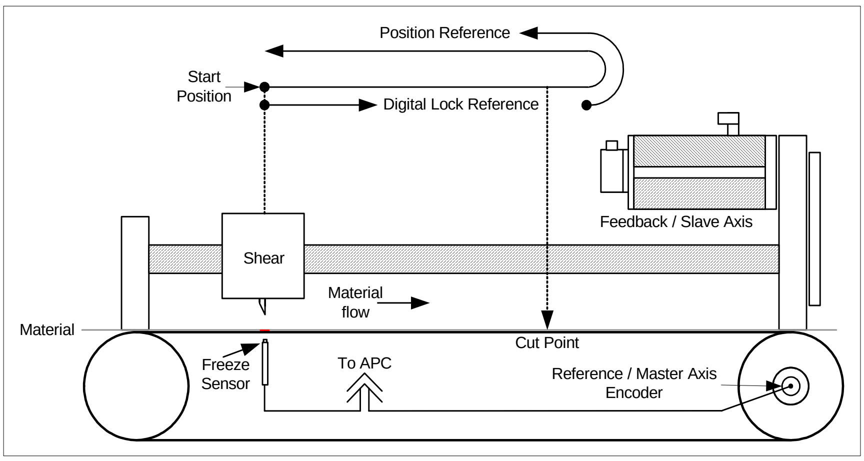
- A Gearbox with a ratio of 2.4:1, would be entered as Numerator 24, Denominator 10.
- A Motor with a 25 tooth pulley connected with a belt to a 60 tooth pulley, would be entered as Numerator 60, Denominator 25.

Where a gear box ratio has a recurring number, or a large number of decimal places it is best to enter the ratio in numbers of teeth, since any remainder from the numerator denominator calculation is retained, therefore giving better speed and positional accuracy.

4.8.6 Digital lock selection by Freeze

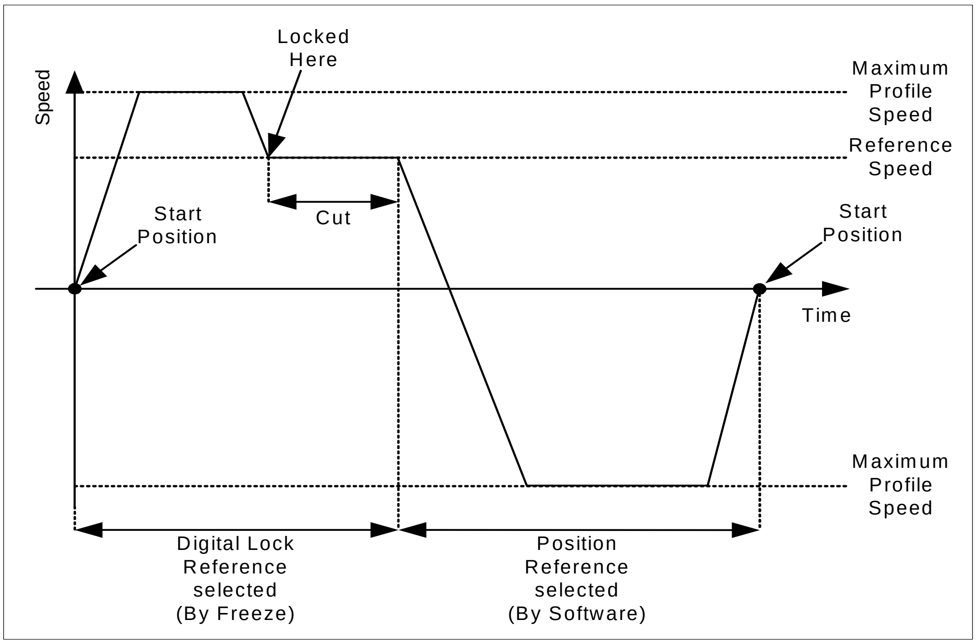
The Digital lock reference may be selected by a freeze input. This is useful when an application requires that a process must be synchronised to a particular position on the Master axis, like a simple flying shear application shown in Figure 4-29 below:

Figure 4-29



This type of application would use Rigid digital lock, such that a mark on the material triggers the freeze input, which in-turn selects the digital lock reference. This will synchronise the Slave axis to the Master axis, at the exact point the freeze occurred. The Digital Lock ratio may needed to compensate for Gear Boxes, and Ball Screw pitch. For this type of simple flying shear to operate the freeze sensor must be aligned with the cutting blade. The cut must only occur once the Master and Slave has become Locked to prevent damage to the cutting blade. Once the cut has been made, the APC must be switched to the Position reference, to return the shear to the start position. The speed profile shown in Figure 4-30 shows movement of the shear.

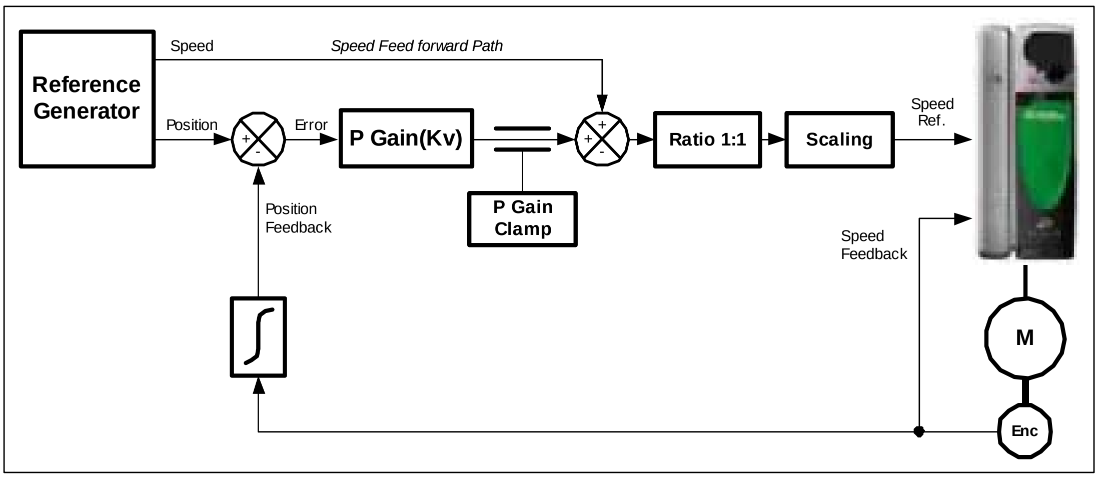
Figure 4-30



4.9 Position Loop

The position loop is used to regulate the position following error to a minimum for a given profile. The drive speed and current loops, when correctly set up, will provide the dynamic response and speed regulation required for the given speed profile, derived by the position loop. See Figure 4-31 below:

Figure 4-31



The position loop consists of two paths: -

- A speed feed forward path, which is derived from the generation of the position reference and provides the main speed reference to the drive e.g. Profile generator or the differential of the following reference position.
- A position correction path, which trims the main speed feed forward path to correct any positional error generated. The proportional gain, is use to amplify the amount of error correction. With a high gain the system will be more responsive, and with a

lower gain less responsive. Too high a gain could increase noise on the motor and also increase instability. With no proportional gain, an increasing following error will occur.

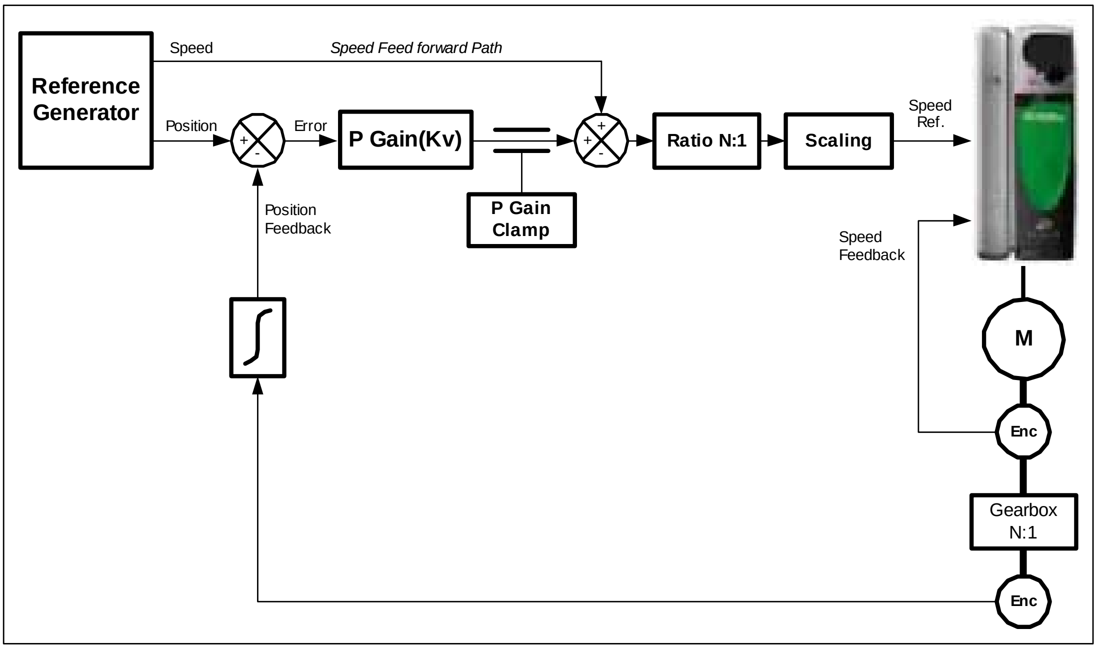
An output clamp is available to restrict the amount of proportional gain speed trim. It is important to ensure the clamp is not set too low as it will restrict the response. Typically the proportional speed clamp is set to 10%, of max speed.

NOTE It is important to ensure the drive speed and torque loops are set correctly for the required bandwidth before the position loop Proportional gain is set.

4.9.1 Output Ratio

The output ratio is in the form of a numerator and denominator and is used to compensate for gearbox ratios, where the feedback position is not fitted and not related to the actual motor speed the drive is controlling. See Figure 4-32 below:

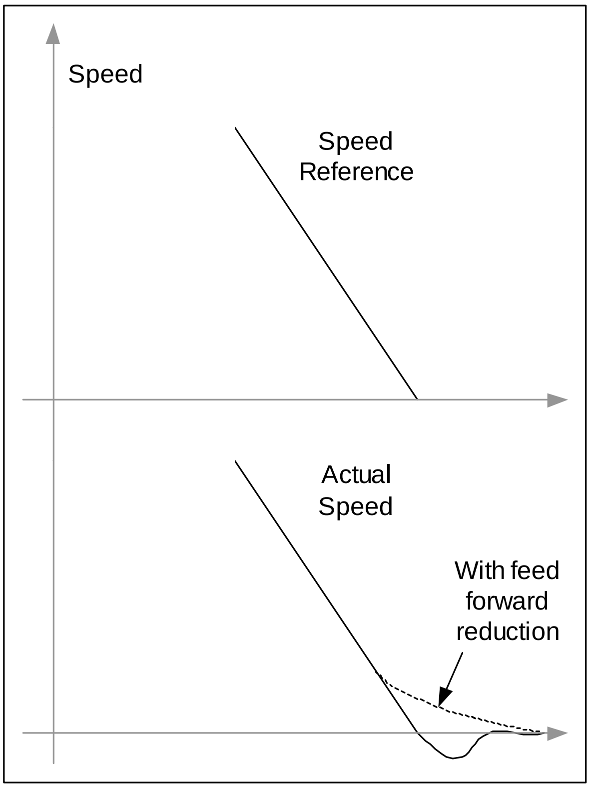
Figure 4-32



4.9.2 Speed Feed Forward Gain

The speed feed forward gain should normally be set to 1, however the gain can be reduced to soften the speed response when coming into final position. This is useful on low bandwidth applications, where no s ramps are used for the acceleration, or the application has a high inertia. See Figure 4-33 below

Figure 4-33



4.9.3 Output Speed Clamp

The output speed clamp should be set to the same speed as drive parameter #1.06. where:

Output Speed Clamp = 2^{32} * #1.06 / 240000

or in DPL:

APCSetOutputSpeedClamp(MULDIV(1073741824, #1.06, 60000)

4.9.4 Torque Feed forward

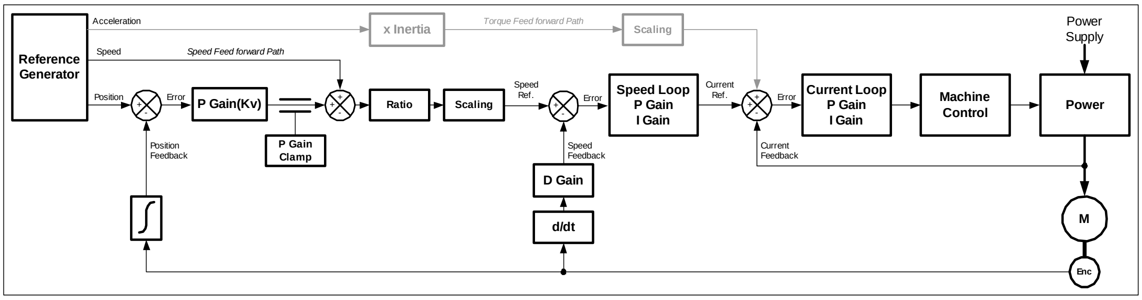
To reduce the following error during acceleration, a torque feed forward term can be generated within the user code. The acceleration of the position reference is readily available on the output of the profile generator. Scaling this from APC units to rad/s/s and when multiplied by the load inertia, the acceleration torque feed forward value can be derived. To use this with the Unidrive-SP, this torque value must be converted into percentage torque as follows:-

%torque = (Acceleration torque * 100) / Full load torque

This value can be summed with the output torque demand from the output of the speed loop, by setting the torque mode selector 04.11 = 4 (speed and torque), and writing the percentage acceleration torque to the torque reference virtual parameter 91.04. See Figure 4-34 below:

NOTE Fast write parameter 91.01 bit 2 must be set to ensure 91.04 is updated with the speed loop update of the drive.

Figure 4-34



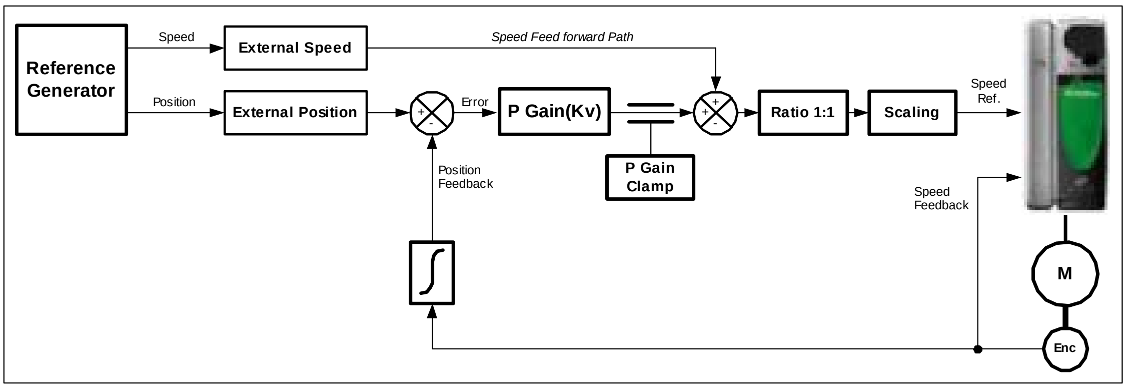
4.9.5 External Speed and Position References

In cases where the APC standard references are not required, the user can generate their own position reference, within POS0 user code and write directly into the position loop external speed and position references. The user must provide the correct values for both references that are generated from same sample period; failing to do this will result in a positional following error.

Example methods of generating external references: -

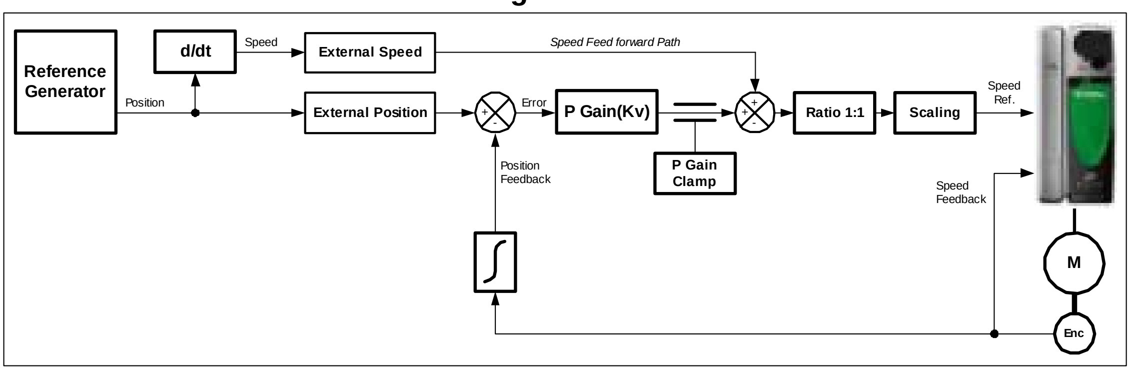
- Generating Speed and Position values directly from reference generator e.g. Profile generator. See Figure 4-35 below:

Figure 4-35



- Generating Speed reference (Feed forward) from the Position reference e.g. Digital lock or electronic gearing. See Figure 4-36 below:

Figure 4-36



NOTE The external speed is in 2^{32} counts per revolution encoder over 250µs sample.

4.10 CTSync

4.10.1 Overview

The SM-Applications module may be used to synchronise two or more Unidrive SP drives. This will ensure that the drives run their internal functions at exactly the same frequency and time, meaning all actions are performed at the same instant.

One (and only one) SM-Applications module must be configured as the Master, and all others configured as Slaves. The Master generates reference data which is transmitted to all Slaves on the network. The Master can be set to operate as a Slave, if for instance two drives need to be synchronised. In this case the Master will be generating the reference data as well as following that reference data. The Slave will also be following that reference data.

4.10.2 Connections

The SM-Applications CTSync operates via a connection between the RS485 ports of the SM-Applications modules on the network in either 2-wire or 4-wire. Refer to the SM-Application manual for information on how to connect the SM-Applications' RS-485 ports.

To simplify wiring the Slave transmit and Master receive signal line, connections can be omitted in 4-wire mode. This is because the Master does not receive a response from the Slave. Therefore only 2 conductors (excluding shield) are required.

The CTSync Master can be changed dynamically in 2-wire mode.

4.10.3 Limitations

- Only one CTSync Master is permitted on the Network
- All CTSync Master and Slave POS tasks must be set to the same update time (parameter #81.12)
- 8 nodes maximum for 2-wire and 16 nodes maximum for 4-wire before line repeaters are required.
- Maximum cable length of RS485 network for CTSync is 200 metres.

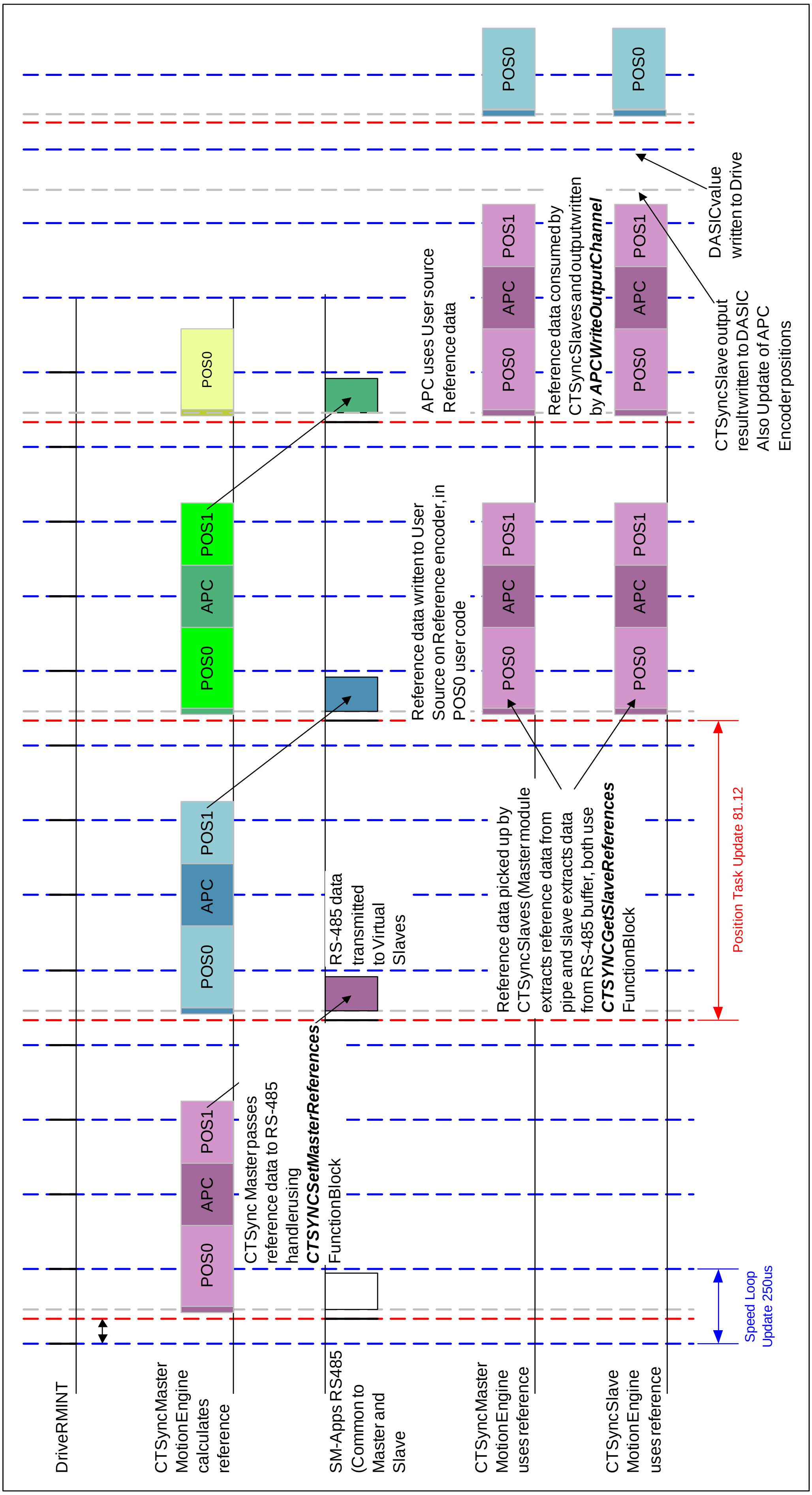
4.10.4 Motion Engine

The tasks performed for each motion engine sample are shown below:

1. The Master motion engine calculates the reference.
2. The Master passes the reference data to the RS-485 handler by using the CTSyncSetMasterReferences function block.
3. The RS-485 data is transmitted to the Slave(s).
4. The reference data is retrieved by the Slave(s), including the Master (if relevant), using the CTSyncGetSlaveReferences function block. This should be performed in the POS0 user code. The reference data can be written to one of the following references:
    - External speed and position references of the APC position loop. This data will be updated within the same POS task cycle the data was retrieved.
    - The User Program encoder source. This will be used on the following POS task, as the APC encoder positions are updated just before the POS task cycle.
5. The reference data is outputted by *APCSetupOutputChannel()* or *APCWriteOutputChannel()* (if required) to the channels specified.
6. The Slave output values are written to the Drive parameters via the Drive's ASIC.

For more information on the timings refer to Figure 4-37 *CTSync and APC Motion Engine*

Figure 4-37 CTSync and APC Motion Engine



4.11 APC Output Channel

When an SM-Applications module writes data to a drive parameter, it does so via a multi-port RAM interface. A write to a drive parameter is handled as a write to a multi-port RAM location, which the Unidrive SP then reads and updates the associated parameter. This is handled every 250μs by the Unidrive SP. There can therefore be up to 250μs between the time at which the SM-Applications module writes a parameter to the multi-port RAM, and when the Unidrive SP transfers the value into the parameter.

This can cause problems if it is not handled correctly. In the case where the SM-Applications module writes its data to the multi-port RAM just before the Unidrive SP reads it out into the parameter, the delay is very small, but if the SM-Applications module was to write to the multi-port RAM just after the Unidrive SP had performed the transfer to the parameter, it would take very nearly 250μs for the drive to check again, and see that new data had been written, and update the associated parameter. This may be a problem when writing to parameters which directly control the drive's output e.g. the Hard Speed Reference. Any jitter at the point when the SM-Applications module user program wrote to the Hard Speed Reference, might result in much larger (integer multiples of 250μs) jitter in the data being used by the drive to control the motor.

A system has been provided to the author of User Programs for the SM-Applications module to ensure the data is written to the multi-port RAM at a known, and constant point in relation to when the Unidrive SP extracts data from the multi-port RAM, and transfers it to the corresponding drive parameter. In this system the user passes the data to be written to a "holder", and this holder, which resides at a low level in the operating system, is capable of ensuring that the transfer to multi-port RAM is handled at the correct time. The time at which the data is transferred to the multi-port RAM is immediately prior to the next motion engine cycle being started. i.e. immediately before POSx task. This means that the data written during the n-th POS task will be written to the multi-port RAM at the very start of the n+1th POS task, and the Unidrive SP will pass this to the associated parameter 250μs later. This means that no matter how long the POS task takes to calculate the data to be written to the drive parameter (as long as the task does not over-run) it will reach the parameter at the same time, with no jitter.

The "holder" for the data to be written, and the information concerning which parameter the data should be written to, is known collectively an "Output Channel".

Although it is quite possible for the user program to write directly to a parameter, this might provide rough position or speed control depending on conditional execution within the POSx tasks, whereas routing the data through an output channel ensures smooth, deterministic operation.

The APC output channel can be configured in two ways:

1. Automatic Configuration - where once configured the APC will continue to write the output speed from the position loop to one of the following destinations:
   - The Unidrive SP Hard Speed Reference drive parameter 3.19
   - The Unidrive SP preset Speed reference parameter 1.21
   - Any un-protected Unidrive SP or SM-Applications module parameter
2. Manual configuration - Once configured this will enable the user to source the reference data to on of the above destinations. This means that the user must write code within one of the POS tasks. This can be used with the read output speed command, *APCGetOutputSpeed()*, where the user can manipulate the output speed, and send it to the drive destination parameter, using the write output command, *APCWriteOutputChannel()*. This should be done in the POS1 task, as this runs after the APC is executed.

The APC output channel may be enabled and disabled whilst the program is running.

In summary, the user has the option of how the APC output reference is written to the destination parameter within the drive. The method chosen will be dependant on the APC's application. For Example:

- For a stand alone, single axis applications, the output reference is required to be written to the destination parameter synchronously with the drive speed loops, therefore simply writing to the destination parameter in user code within the POS1 task is sufficient, although the output channel method can also be used.
- For synchronised, multi slave axis applications using CTSync, it is important to ensure the output channel method is used, as this will ensure all the slave APC output references, are updated at the same time, therefore minimising slave to slave positional jitter.

In firmware version =>01.04.xx, four additional modes have been added which allows the user to invert the speed reference sent to the drive:

- 10 = Hard Speed Reference Inverted
- 12 = Speed Reference (#1.21) Inverted
- 13 = User Parameter Selection Inverted
- 14 = User Reference Inverted

These are useful if the user needs to invert the Feedback encoder data. Using the inverted mode corrects the position loop output direction to match the feedback inversion.

4.12 APC Operation in User Units

The APC has some additional features added to enable the APC to operate directly in user units when the SM-Applications is running in PLCopen mode. This change occurred because PLCopen is really a user interface layer, which controls the APC.

In SM-Applications firmware V01.04.04, and Sypt Pro V02.00.11, the new functionality has been made available to APC users. This new functionality allows the user to:

- Enter and read position in user units, with remainder to prevent loss of position information
- Enter and read speed in user units per second
- Enter and read acceleration / deceleration in user units per second per second
- No need to convert externally using the APCToUser and UserToAPC conversion method (saves an additional command call)
- All positioning references may be controlled in user units (except the CAM reference which will be implemented soon.

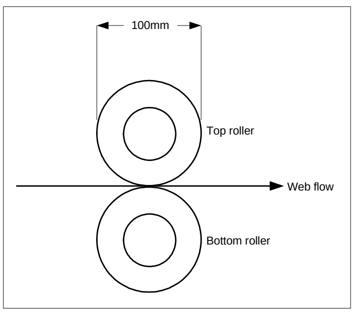
User units are whatever unit of position measurement a system is running in e.g. mm, inches, bottles, sheets etc. If for example mm are the chosen position unit, then speeds will be in mm/s, and acceleration will be in mm/s/s.

4.12.1 Setting up a User Unit conversion ratio

The user unit conversion function is disabled by default, so that old code will still operate correctly i.e. positions in encoder counts, speeds in 2^{32} encoder counts per rev / 250μs. To change this, a new command SetUPRNumeratorAndDenominator(N%,D%) has been introduced, where N% = the number of user units, and D% = the number of revolutions for the number of user units (**units per rev**), e.g:

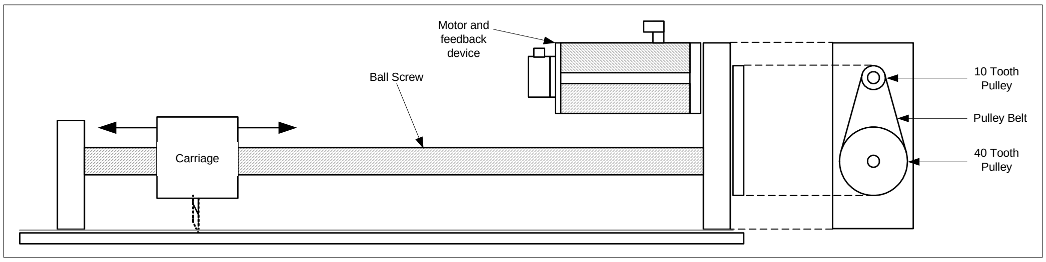
1. A nip roller feeding an application has a diameter of 100mm. To run the roller at a line speed defined in mm/s the circumference of the nip roll will be π * 100 = 314, or 314mm per revolution of the roller. This would give a numerator (N%) of 314, and a denominator (D%) of 1. If a greater ratio accuracy is required e.g. 4 decimal places the numerator would be 3141592, and the denominator would be 10000.

Figure 4-38



2. A motor with a 4:1 pulley system, is driving a ballscrew carriage with a turn pitch of 10mm per revolution. To run the carriage in mm position increments, the numerator = 10 and the denominator = 4. If base units of 1mm are too coarse, they can be changed to microns by setting a numerator of 10000 and a denominator of 4.

Figure 4-39



The numerator and denominator may each be set within a range of 1 to 2147483647. The value of the numerator and denominator are used to set the APC's internal conversion numerator and denominator [96] and [97], where:

[96] = $2^{(32-[14])}$ * D%

[97] = N%

4.12.2 Reading parameters in User and APC Units

The function of APCReadPar(Par%) has been modified to include automatic user unit conversion, when a conversion ratio is set. This change affects all position, speed and acceleration parameters, except those specifically related to the CAM reference. This will allow the user to read values rounded to the nearest user unit.

If the nearest user unit is not accurate enough for the application, and the exact position is required e.g. homing etc., a new read parameter command called APCReadParWithRem(Par%) has been created, which will gives the user the ability to read a position parameter to the nearest user unit with a signed remainder. Only position parameters will return a remainder.

NOTE **The converted units are rounded to the nearest whole unit, therefore the remainder is signed. A positive remainder indicates that the whole user unit value was rounded down, and a negative remainder indicates that it has been rounded up.**

If the user needs to read a raw APC unit value when a conversion ratio is set, 1000 can be added to the read parameter of interest, using APCReadPar(Par%) to obtain the value e.g. the feedback main position, [48], becomes [1048], and will be read in APC units. **APCReadParWithRem(Par%) can not be used with +1000 read parameters.**

The following read parameters are permanently excluded from conversion when read, as they are "raw" APC values used by the APC:

[31] [51] [170] [171] [172] [173] [174] [175] [176] [178] [180] [181] [182] [183] [184]

4.12.3 Setting positions with remainder

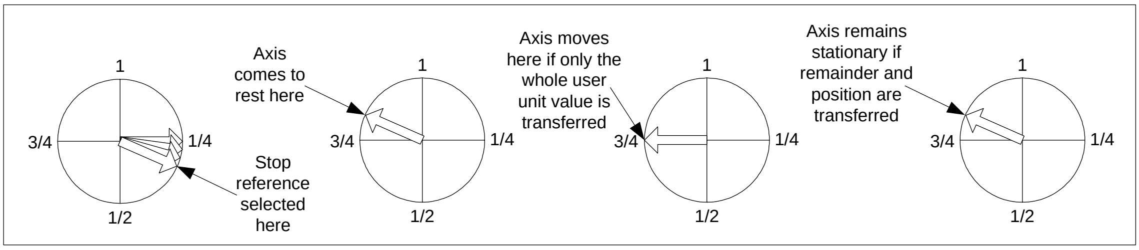
The APC Position setpoints [92] and [93] now have a remainder input [172] and [175], for the main position input and offset position input respectively. The following commands are used to set the main and offset position setpoints with remainder:

- To set the main position setpoint use APCSetPositionSetPoint(Position%) and APCSetSetPointRem(Remainder%)
- To set the offset position setpoint use APCSetPositionOffset(Position%) and APCSetOffsetRem(Remainder%)

A remainder is required for accurate positioning in user units, because the APC uses integer maths, therefore without a remainder any conversions to or from user units would lose accuracy, as fractional part would be lost, e.g:

If the user unit conversion numerator is 4 and the denominator is 1 the system will run in quarter of a revolution units. If the system was running at 2 unit /s, and then the stop reference is selected with a decel rate of 1unit /s/s, the axis will stop 2 user units later (see Figure 4-40). If only the whole user units value is transferred to the position reference, and the position reference is selected, the axis will move back to the 3/4 rev position, instead of staying stationary (see Figure 4-40). This is because any position past a whole user unit is represented by the remainder. If the remainder and the whole user unit value is transferred, the axis will remain stationary.

Figure 4-40



NOTE **The remainder must be set before the user unit value, when setting user unit values in a position reference e.g. APCSetSetPointRem(Remainder%) then APCSetPositionSetPoint(Position%). This is because setting the user unit value triggers the in-bound conversion of the user units.**

4.12.4 Calculating positions in User Units with remainder

To convert positions in APC units (encoder counts) to user units with remainder the following formulae may be used:

UU	= User Units
Rem	= Remainder
APC Pos	= APC Position in encoder counts
UPR N	= Units per revolution numerator
UPR D	= Units per revolution denominator
[14]	= Number of turns bits as shown by read parameter [14]
UU	= Round(APC Pos * (UPR N/($2^{(32-[14])}$ * UPR D)))
Rem	= (APC Pos * UPR N) - (UU * ($2^{(32-[14])}$ * UPR D))

4.12.5 Homing routines in User Units

There are two basic types of Homing routine that can be created using the APC:

1. Home to a captured position (Freeze or Marker capture)
2. Home to a parameter (rising or falling edge)

For homing routines of type 1 the sequence should be:

- Run the axis at some speed searching for Freeze or Marker
- Marker or freeze flag goes high
- Read the marker or freeze position using APCReadParWithRem(Par%)
- Write the new position and remainder to the main position reference using APCSetPositionSetPoint(Position%) and APCSetSetPointRem(Remainder%) respectively, and add a homing offset to the position setpoint if required.
- Wait till the new position is reached

- An appropriate delay e.g. 0.5s, to ensure there is no positional following error
- Reset the feedback main position counter, [48], with offset if required
- Check the feedback main position counter, [48], has been reset

For homing routines of type 2 the sequence should be:

- Run the axis at some speed searching for a change in the homing parameter (connected to a digital input, set by a PLC, set by a current threshold etc.)
- When the parameter changes, reverse the direction of rotation, and reduce the speed
- Wait till the original state of the homing parameter is read
- Read the feedback position [48] using APCReadParWithRem(Par%)
- Write the new position and remainder to the main position reference using APCSetPositionSetPoint(Position%) and APCSetSetPointRem(Remainder%) respectively, and add a homing offset to the position setpoint if required.
- Wait till the new position is reached
- An appropriate delay e.g. 0.5s, to ensure there is no positional following error
- Reset the feedback main position counter, [48], with offset if required
- Check the feedback main position counter, [48], has been reset

4.12.6 Changing from the Stop reference to the Position reference

To assist the user when changing from the Stop reference to the Position reference without causing the axis to move, the position reference must be set to the same value as the stop position [91], and the stop position remainder [178]. If the whole user unit value [91] was set in the Position reference, without setting the remainder, the axis could rotate to the nearest whole user unit instead of remaining stationary. To avoid this, APCTransferStopPosToPosSetPoint() has been created, to pass the Position along with any remainder directly to the Position reference, thus avoiding unexpected moves when the position reference is selected.

4.12.7 Inserting positional filters

In SM-Applications firmware version V01.03.03, the user was given additional commands to allow the insertion of filters or scaling to the reference and feedback encoder counters. This is done by:

- Disabling the encoder counter input e.g. APCDisableRefInput
- Read the source counter e.g. APCReadPar(20), and use the value as the input to the filter
- Take the output from the filter, and write the new value back in to the encoder counter input e.g. APCSetRefInput(Value%)

When user unit mode is selected by setting a units per rev numerator and denominator, this operation has to be handled slightly differently. The reference and feedback inputs are always set in encoder counts, however when the encoder counter source value is read using APCReadPar(Par%), it will be converted to the nearest whole user unit value not a “raw” encoder count value, giving a mismatch in data types. To read the value in encoder counts 1000 must be added to the encoder source counter read parameter number e.g. [20] becomes [1020], and [40] becomes [1040]. See the example below:

- Disable the encoder counter input e.g. APCDisableRefInput
- Read the source counter e.g. APCReadPar(1020), and use the value as the input to the filter
- Take the output from the filter, and write the new value back in to the encoder counter input e.g. APCSetRefInput(Value%)

5 APC Command Descriptions

5.1 APC Functions

The Advanced Position Controller has different built-in user configured motion functions; these are as follows:

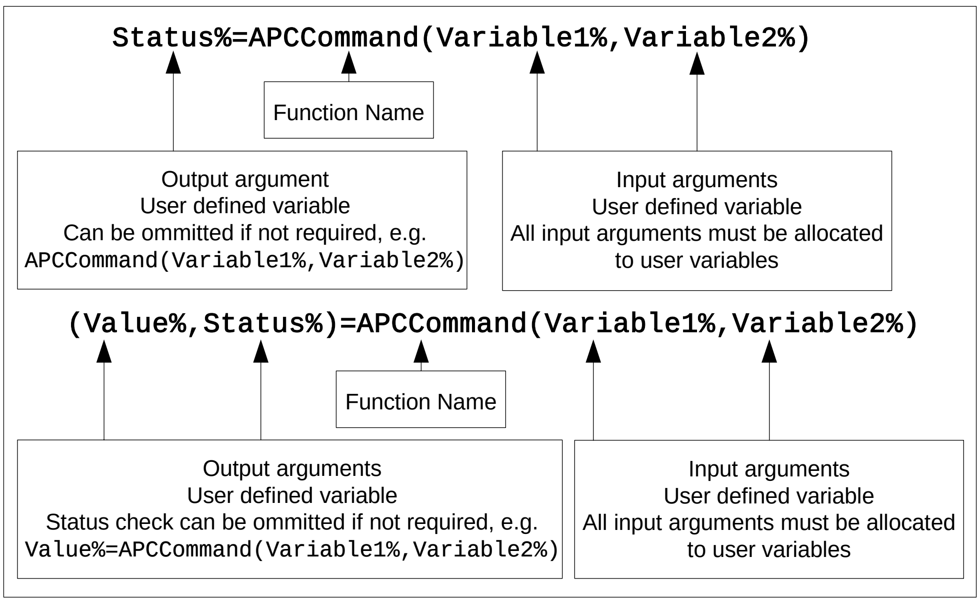
1. Digital Lock (electronic gearbox).
2. CAM
3. User defined position and speed setpoints
4. Linear speed profile generator
5. Closed-loop position controller

5.2 DPL Commands

Apart from the SM-Applications setup parameters (menu 15, 16 or 17), no other parameters or registers need to be configured to run the APC. All configurations and operations are actioned by DPL function calls, using the DPL Editor in Sypt Workbench.

Each APC DPL function call may have input or output arguments. It is important that the argument position is considered and retained, when changing variable names to ensure correct functionality of the command. See Figure 5-1 :

Figure 5-1



The status check lets the user know whether a command has been successfully executed or not. This is useful for debugging programmes, but may not be required for final code. The status output can take one of two forms:

1. Boolean - where 0 = Operation Failed and 1= Operation Successful
2. Numerical - where 0 and 1 are the same as Boolean, however there are other numbers for specific reasons for an Successful operation. Consult the individual command description, for a full list of the status returns for any specific command.

The following Section gives detailed descriptions of each APC function command, and should be used in conjunction with the *Functional Description* section.

5.2.1 Control and Access Functions

APCSetRunMode		
`Status% = APCSetRunMode(Mode%)`		
Input Arguments	*Mode%*:	
	0:	The APC is not called.
	1:	The APC will be called, but only the Reference and Feedback counters will be updated.
	2:	The APC will run and perform the position control, CAM, and digital lock operations as configured.
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description	*Mode%*	
	0	With `Mode%` set to 0, the APC is completely disabled and has no effect.
	1	With `Mode%` set to 1, the APC runs in the disabled state, and only allows the Reference and Feedback counters to be used; no other functions of the APC are available. **Behaviour of the Reference and Feedback Integrated Counters [28] and [48]:** The reference and feedback integrated counters will be held in their reset state (value depends on whether relative or absolute mode is selected), when *APCResetSourcesOnDisable* is active; refer to Table 4-1. Otherwise, if *APCDoNotResetSourcesOnDisable* is active, then the APC integrated position counters will increment or decrement, with respect to the direction of the source position. The APCReset() has no effect in this mode. **To Ensure that the APC Counters and parameters are initialised correctly on power up, APC run mode 1 must be called in the Initial task, and at least 1 POS task cycle must have occurred before using the counter values.**
	2	With `Mode%` set to 2, the APC is fully functional and will perform as configured. When mode 2 is selected, the internal integrated position counters are set as shown in Table 4-1. **Behaviour of the APC Integrated counters:** The integrated position counters (feedback, reference, and profile references), can be reset in two ways: - 1. Forcing the APC run mode to disable (1), providing *APCResetSourcesOnDisable* is active. 2. Using dedicated APC function *APCReset().* The feedback and Profile counters, [48], [115], and [117], can be reset with or without a offset defined by *APCSetPositionResetOffset.* Where *APCDoNotResetSourcesOnDisable* is active, an *APCReset()* will have no effect on the counters. **To Ensure that the APC Counters and parameters are initialised correctly on power up, APC run mode 1 must be called in the Initial task, and at least 1 POS task cycle must have occurred before using the counter values. See section 4.3.1** *Enabling the APC* .

APCSetRunMode	
`Status% = APCSetRunMode(Mode%)`	
APCReadPar	[0] and [4]
Units	NA
Range	0 to 2
Default	0

APCReset		
`Status% = APCReset()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command is only functional when the APC run mode is set to 2, *APCSetRunMode(2).* When *APCResetSourcesOnDisable* is active, using this command will reset the integrated position counters[28], [48], [115], and [117] to the absolute or relative positions, if there is a reset offset position value, defined by *APCSetPositionResetOffset,* this will be added to the feedback/profile integrated counters. Where *APCDoNotResetSourcesOnDisable* is active, APCReset has no effect on the integrated counters, but will be able to restart the CAM in single shot mode. When ever this command is used the APC will run for one cycle in Disabled mode in order to reset the integrated counters [28] and [48]. In most homing applications it is important to know if the APC has finished resetting or not, and to facilitate this read parameter [5] has been introduced. When [5] = 1 a reset is in progress, and when [5] changes from 1 to 0 the reset is complete. Only available in SM-Applications firmware =>V01.04.xx.
APCReadPar		[5]
Units		NA
Range		Active or Inactive
Default		Inactive

APCGetOutputSpeed		
`Speed% = APCGetOutputSpeed()`		
Input Arguments	None	
Output Arguments	*Speed%*:	Output speed
Description		This command will return the value of the Position loop output [135], and load the result into a user variable; in this case `Speed%`. Where the recommended output channel command is not used, *APCSetupOutputChannel()*, this command can be used as an alternate method of sourcing the output speed reference to the drive by the user program. Note: When the drive speed reference is sourced from the user program, it will get updated on the next 250µs of the drive speed loop. This is not recommended for high precision, synchronised multiple slave axes.
APCReadPar		[135]
Units		2^{32} counts / rev encoder over 250µs or user units/s
Range		+/- Output Speed Clamp [140]
Default		0

APCGetOutputSpeedRpmx10		
`Speed% = APCGetOutputSpeedRpmx10()`		
Input Arguments	None	
Output Arguments	*Speed%*:	Output speed converted to 0.01 rpm units
Description		This command will return the value of the Position loop output [135], converted into 0.01rpm units, and load the result into a user variable; in this case `Speed%`. Where the recommended output channel command is not used, *APCSetupOutputChannel()*, this command can be used as an alternate method of sourcing the output speed reference to the drive by the user program. Note: When the drive speed reference is sourced from the user program, it will get updated on the next 250µs of the drive speed loop. This is not recommended for high precision, synchronised multiple slave axes.
APCReadPar		[135]
Units		rpm x 10
Range		+/- Output Speed Clamp [140]
Default		0

APCSelectAbsoluteMode		
`Status% = APCSelectAbsoluteMode()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		When this command is active the integrated position counters will be preset with the corresponding source position after power up, or an APCReset.The reset offset position value, defined by *APCSetPositionResetOffset*, will only be added to the feedback/ profile integrated counters. This will be a true absolute position where an absolute encoder is used (e.g. SinCos, SSI). The counters can be preset as follows: - - In APC Run Mode 1 *APCSetRunMode(1)* and *APCResetSourcesOnDisable* Integrated position counters [28], [48] only. - In APC Run Mode 2 *APCSetRunMode(2)*, *APCResetSourcesOnDisable()*, and *APCReset()* Integrated position counters [28], [48],[115], [117]. The position counters [20] and [40] are a duplication of the source counters with the set user resolution defined by *APCSetNumOfTurnsBits(TurnBits%)*.
APCReadPar		[1]
Units		NA
Range		Active or Inactive
Default		Inactive

APCSelectRelativeMode		
`Status% = APCSelectRelativeMode()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		When this command is active the integrated position counters will be reset to zero after power up, or an APC reset.The reset offset position value, defined by *APCSetPositionResetOffset*, will only be added to the feedback/profile integrated counters. The counters can be reset as follows: - - In APC Run Mode 1 *APCSetRunMode(1)* and *APCResetSourcesOnDisable* Integrated position counters [28], [48] only. - In APC Run Mode 2 *APCSetRunMode(2)*, *APCResetSourcesOnDisable()*, and *APCReset()* Integrated position counters [28], [48],[115], [117]. The position counters [20] and [40] are a duplication of the source counters with the set user resolution defined by *APCSetNumOfTurnsBits(TurnBits%)*.
APCReadPar		[1]
Units		NA
Range		Active or Inactive
Default		Active

APCReadPar		
`(Value%, Status%) = APCReadPar(APCParameterNumber%)`		
Input Arguments	*APCParameterNumber%:*	The index number for the required parameter
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
	Value%:	Shows the value of the APC parameter specified by APCParameterNumber%.
Description		This command allows the user to read any of the parameters within the APC (shown in diamonds on the APC Logic Diagram or square brackets[ ] in text), and write the contents into a DPL variable. If a user unit conversion ratio is set using *SetUPRNumeratorAndDenominator(N%,D%)* APCReadPar may be used to read the APC units value by adding 1000 to the parameter number. see section 4.12 for more details
Units		Dependant on parameter read
Range		Dependant on parameter read
Default		Dependant on parameter read

APCReadParWithRem		
`(Value%,Remainder%, Status%) = APCReadParWithRem(APCParameterNumber%)`		
Input Arguments	*APCParameterNumber%:*	The index number for the required parameter
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
	Value%:	Shows the value of the APC parameter specified by APCParameterNumber%.
	Remainder%:	If APCParameterNumber% is directed to a positional parameter e.g. [92] or [115] etc., Remainder% will show the value of the positional remainder, when a user unit conversion ratio is set by *SetUPRNumeratorAndDenominator(N%,D%).* If APCParameterNumber% is directed to a non positional parameter or a CAM specific parameter Remainder% will be set to 0. This command is useful when a user unit conversion ratio is set, and an accurate position is required to be set in the position setpoint. This command may not be used with +1000 read parameters. See section 4.12 for more details on the use of this command
Description		This command allows the user to read any of the parameters within the APC (shown in diamonds on the APC Logic Diagram or square brackets[ ] in text), and write the contents into a DPL variable.
Units		Dependant on parameter read
Range		Dependant on parameter read
Default		Dependant on parameter read

APCSetPositionResetOffset		
`Status% = APCSetPositionResetOffset(Position%)`		
Input Arguments	*Position%*:	The position to be used for the offset to the Feedback main position [48] when an APC Reset is called, or the APC is disabled (Run Mode 1).
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command sets the reset offset position, the value of which will be added to the Feedback/Profile integrated position counter value, after power up, or an APC reset. The counters can be reset with offset as follows: - - In APC Run Mode 1 *APCSetRunMode(1)* and *APCResetSourcesOnDisable* Integrated position counters [28] and [48] only. - In APC Run Mode 2 *APCSetRunMode(2)*, *APCResetSourcesOnDisable()*, and *APCReset()* Integrated position counters [28] [48],[115], [117]. Where an offset is applied, there will be no resultant following error. However if the Digital Lock reference is selected by *APCSelectReference(4)*, with *APCEnableRigidLock()* active, the resultant following error will equal the reset offset position value, and will therefore cause a corresponding speed reference to be generated. Read parameter [2] shows the feedback offset value. In SM-Apps Firmware =>V01.04.04 the feedback offset position may be set in user units, by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[2]
Units		Encoder Counts
Range		-2^{31} to $2^{31} - 1$
Default		0

APCSetRefPosResetOffset		
`Status% = APCSetRefPosResetOffset(Position%)`		
Input Arguments	*Position%*:	The position to be used for the offset to the reference main position [28], when an APC Reset is called, or the APC is disabled (Run Mode 1).
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command sets the reference reset offset position, the value of which will be added to the reference integrated position counter [28] value, after power up, or an APC reset. The counters can be reset with offset as follows: - - In APC Run Mode 1 *APCSetRunMode(1)* and *APCResetSourcesOnDisable* Integrated position counters [28] and[48] only. - In APC Run Mode 2 *APCSetRunMode(2)*, *APCResetSourcesOnDisable()*, and *APCReset()* Integrated position counters [28], [48], [115], [117]. Where an offset is applied, there will be no resultant following error. However if the Digital Lock reference is selected by *APCSelectReference(4)*, with *APCEnableRigidLock()* active, the resultant following error will equal the reset offset position value, and will therefore cause a corresponding speed reference to be generated. Read parameter [6] shows the reference offset value. In SM-Apps Firmware =>V01.04.04 the reference offset position may be set in user units, by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[6]
Units		Encoder Counts
Range		-2^{31} to $2^{31} - 1$
Default		0

APCResetSourcesOnDisable		
`Status% = APCResetSourcesOnDisable()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		When this command is active the integrated position counters will be reset to a value, dependant on whether Absolute or Relative mode is selected, after power up, or an APC reset. The reset offset position value, defined by *APCSetPositionResetOffset* will then be added to the feedback and profile integrated counters, and the reference offset value defined by *APCSetRefPosResetOffset*, will be added to the reference integrated counter. The counters can be reset as follows: - - In APC Run Mode 1 *APCSetRunMode(1)* and *APCResetSourcesOnDisable* Integrated position counters [28], [48] only. - In APC Run Mode 2 *APCSetRunMode(2)*, *APCResetSourcesOnDisable()*, and *APCReset()* Integrated position counters [28], [48],[115], [117].
APCReadPar		[3]
Units		NA
Range		Active or Inactive
Default		Active

APCDoNotResetSourcesOnDisable		
`Status% = APCDoNotResetSourcesOnDisable()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		When this command is active the integrated position counters can not be reset, or given an offset, after power up, or an APC reset. This command is solely intended for allowing the user in APC Run mode 1, with Relative mode selected, to use the Reference and Feedback Integrated counters, without the integrated count being reset to zero.
APCReadPar		[3]
Units		NA
Range		Active or Inactive
Default		Inactive

5.2.2 Reference and Feedback Encoder

Figure 5-2

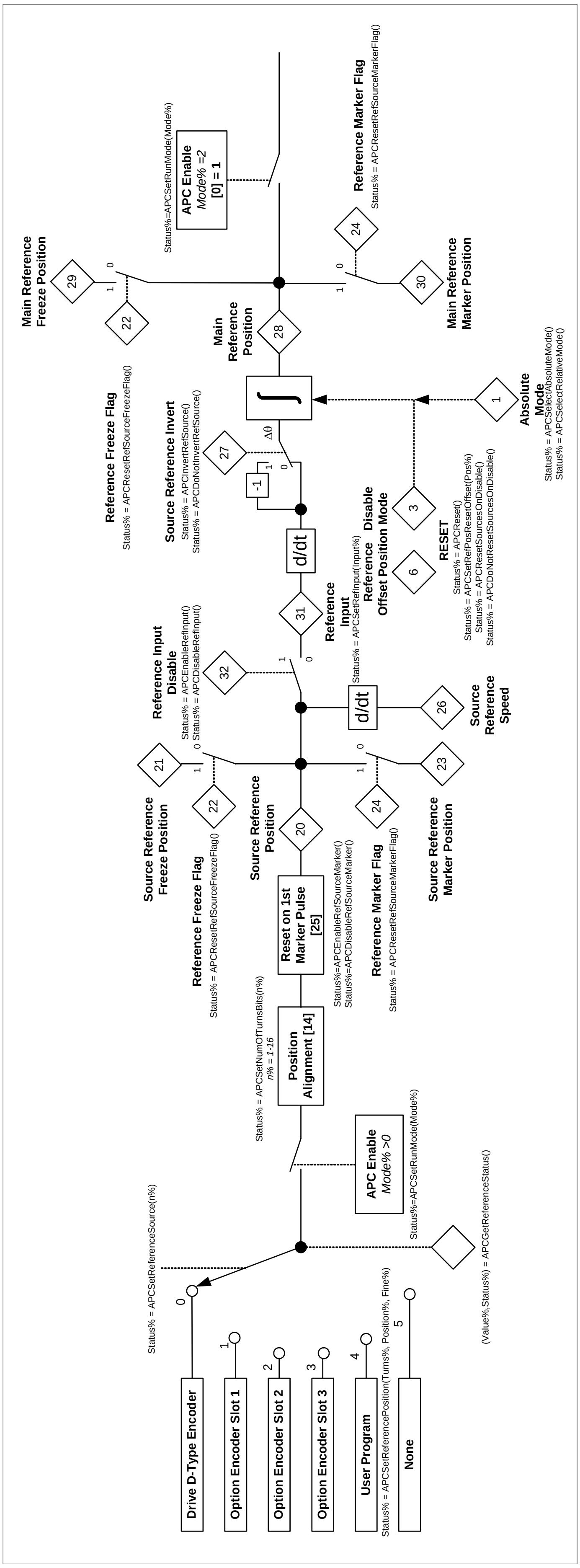
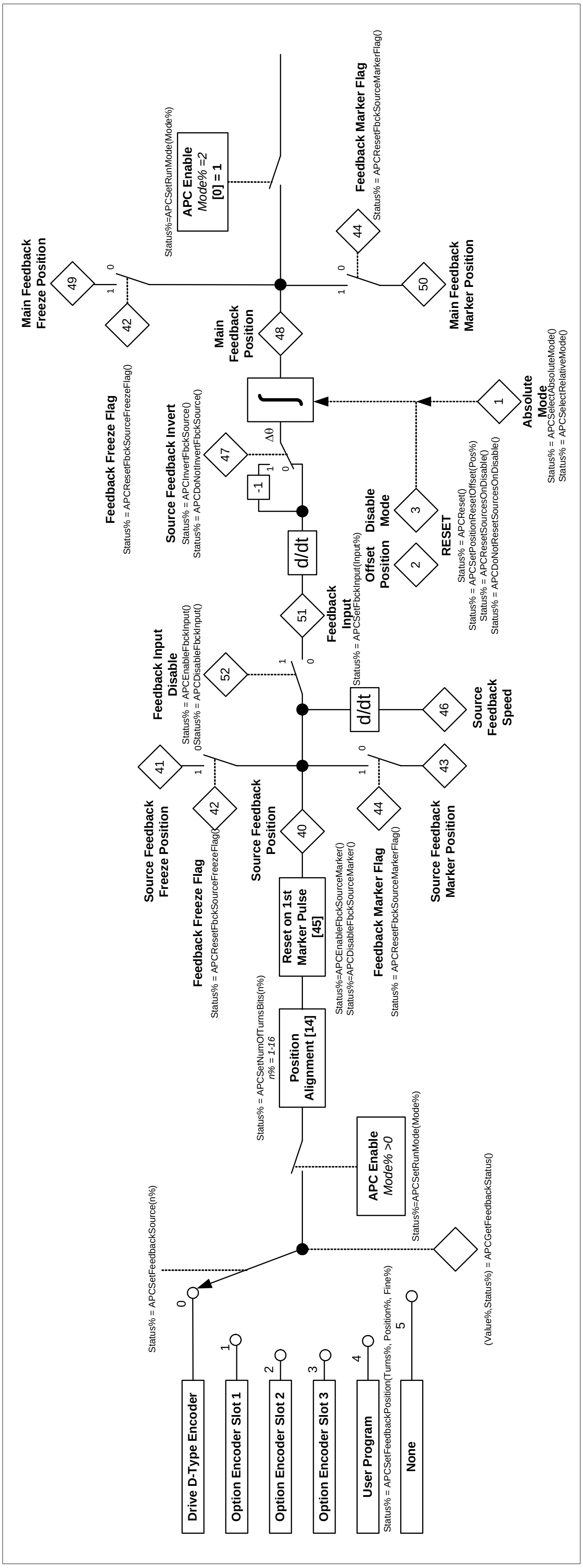
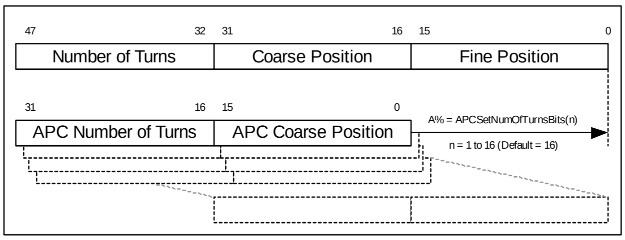


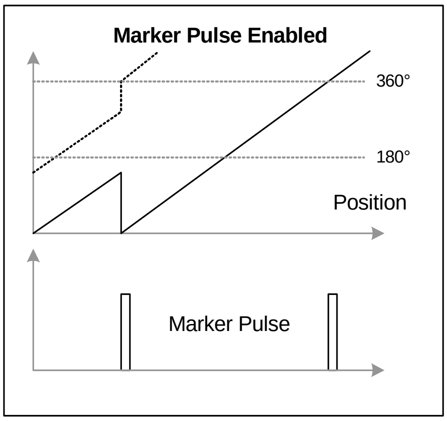
Figure 5-3

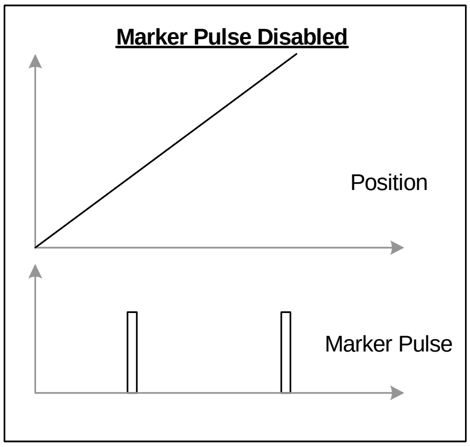

Drive D-Type Encoder
Option Encoder Slot 1
Option Encoder Slot 2
Option Encoder Slot 3
User Program
Status% = APCSetFeedbackPosition(Turns%, Position%, Fine%)
None
0
1
2
3
4
5
Status% = APCSetFeedbackSource(n%)
(Value%,Status%) = APCGetFeedbackStatus()
APC Enable
Mode% >0
Status%=APCSetRunMode(Mode%)
Position Alignment [14]
Status% = APCSetNumOfTurnsBits(n%)
n% = 1-16
Reset on 1st Marker Pulse [45]
Status%=APCEnableFbckSourceMarker()
Status%=APCDisableFbckSourceMarker()
Source Feedback Position
40
Source Feedback Freeze Position
41
Feedback Freeze Flag
42
Status% = APCResetFbckSourceFreezeFlag()
Source Feedback Marker Position
43
Feedback Marker Flag
44
Status% = APCResetFbckSourceMarkerFlag()
d/dt
46
Source Feedback Speed
Feedback Input Disable
52
Status% = APCEnableFbckInput()
Status% = APCDisableFbckInput()
51
Feedback Input
Status% = APCSetFbckInput(Input%)
d/dt
-1
Source Feedback Invert
47
Status% = APCInvertFbckSource()
Status% = APCDoNotInvertFbckSource()
Δθ
Position Offset
2
Disable Mode
3
RESET
Status% = APCReset()
Status% = APCSetPositionResetOffset(Pos%)
Status% = APCResetSourcesOnDisable()
Status% = APCDoNotResetSourcesOnDisable()
Absolute Mode
1
Status% = APCSelectAbsoluteMode()
Status% = APCSelectRelativeMode()
Main Feedback Position
48
Main Feedback Freeze Position
49
Feedback Freeze Flag
42
Status% = APCResetFbckSourceFreezeFlag()
Main Feedback Marker Position
50
Feedback Marker Flag
44
Status% = APCResetFbckSourceMarkerFlag()
APC Enable
Mode% =2
[0] = 1
Status%=APCSetRunMode(Mode%)

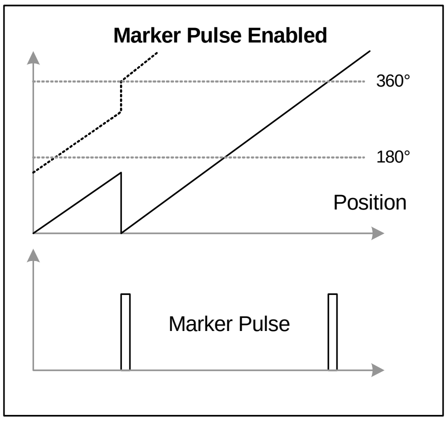
APCSetReferenceSource		
`Status% = APCSetReferenceSource(Source%)`		
Input Arguments	*Source%*:	
	0:	Drive
	1:	Slot 1
	2:	Slot 2
	3:	Slot 3
	4:	User program
	5:	Unconfigured
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
	-3:	Another process is modifying the object's data
Description	*Source%*:	
	0:	With `Source%` set to 0, the Drive encoder input is selected as the Reference Position source.
	1:	With `Source%` set to 1, the feedback option in Slot 1 is selected as the Reference Position source.
	2:	With `Source%` set to 2, the feedback option in Slot 2 is selected as the Reference Position source.
	3	With `Source%` set to 3, the feedback option in Slot 3 is selected as the Reference Position source.
	4	With `Source%` set to 4, Reference Source position can be manually written to the APC using the command *APCSetReferencePosition(Turns%, Position%, PositionFine%).*
	5	With `Source%` set to 5, the Reference Position source is selected as unconfigured. This means that the Reference Source position will be set to zero.
Note		**When this command has been used to set the reference encoder location, the reference encoder virtual parameters in menu #90 will also be directed to the same encoder. The equivalent of this command in menu 90 is 90.43** **This command is intended to be used in the Initial task only, however in applications which require the encoder source to be changed it should be done in the background task, on a one shot basis; under no circumstances should the encoder source be set continually.**
APCReadPar		No read parameter use *APCGetReferenceStatus()* instead
Units		NA
Range		0 to 5
Default		0

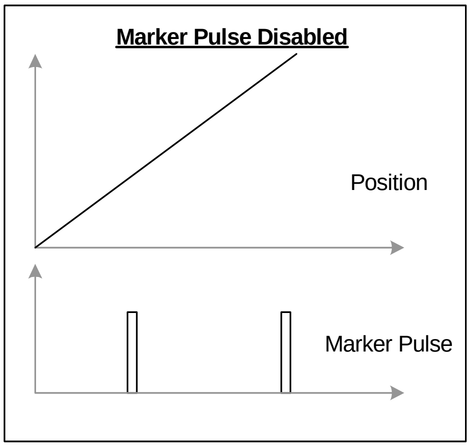
APCSetFeedbackSource		
`Status% = APCSetFeedbackSource(Source%)`		
Input Arguments	*None*	
	0:	Drive
	1:	Slot 1
	2:	Slot 2
	3:	Slot 3
	4:	User program
	5:	Unconfigured
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
	-3:	Another process is modifying the object's data
Description	*Source%*:	
	0:	With `Source%` set to 0, the Drive encoder input is selected as the Feedback Position source.
	1:	With `Source%` set to 1, the feedback option in Slot 1 is selected as the Feedback Position source.
	2:	With `Source%` set to 2, the feedback option in Slot 2 is selected as the Feedback Position source.
	3:	With `Source%` set to 3, the feedback option in Slot 3 is selected as the Feedback Position source.
	4:	With `Source%` set to 4, Feedback Source position can be manually written to the APC using the command *APCSetFeedbackPosition(Turns%, Position%, PositionFine%).*
	5:	With `Source%` set to 5, the Feedback Position source is selected as unconfigured. This means that the Feedback Source position will be set to zero.
Note		**When this command has been used to set the Feedback encoder location, the Feedback encoder virtual parameters in menu #90 will also be directed to the same encoder. The equivalent of this command in menu 90 is 90.44** **This command is intended to be used in the Initial task only, however in applications which require the encoder source to be changed it should be done in the background task, on a one shot basis; under no circumstances should the encoder source be set continually.**
APCReadPar		No read parameter use *APCGetFeedbackStatus()* instead
Units		
Range		0 to 5
Default		0

APCSetNumOfTurnsBits		
`Status% = APCSetNumOfTurnsBits(TurnBits%)`		
Input Arguments	*TurnBits%*:	Value in range 1 to 16 inclusive
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command sets the APC resolution, by defining the number of turns bits and position information seen by the APC. The APC views the position information available from a 48 bit position word through a 32 bit window. The 48 bit word is made from 16 bits of turns (upper 16), 16 bits of coarse position (middle 16), and 16 bits of fine position (lower 16). From default, the APC uses a 32 bit position word made from 16 bits of turns information (upper 16 bits), and 16 bits of coarse position information (lower 16 bits). When the number of turns bits is set below 16, the APC's 32 bit window is shifted right by the same number of bits e.g. If the number of turns bits is set to 10, then the 32 bit position window used by the APC will comprise of 10 turns bits, 16 coarse position bits and 6 fine position bits, giving a total of 1024 turns counts (2^{10}), and 4194304 position counts / revolution ($2^{16 + 6}$). If the chosen encoder, has a resolution of 16 bit or less, altering the number of turns bits will not give extra positional information per revolution, it will only reduce the number of turns bits seen by the APC. 
APCReadPar		[14]
Units		NA
Range		1 to 16
Default		16

APCEnableRefSourceMarker		
`Status% = APCEnableRefSourceMarker()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command enables the reference source marker to reset the reference position within one revolution, so that zero position and the marker pulse are aligned:.  For this to happen, the reference marker flag must be enabled using the command *APCEnableReferenceMarker()*, and the reference marker flag must be reset by using the command *APCResetRefSourceMarkerFlag()* [24]. If the motor has to turn less than 180° before the marker is found, then the reference position counters will jump down to zero i.e. zero position turn 0. If the motor has to turn more than 180° before the marker is found, then the reference position counters will jump up to zero i.e. zero position, turn 1. The turns count is not reset to zero after a marker reset. Once the position counters are aligned this function will have no effect. A Reference Marker Position capture is not possible when this command is active.
APCReadPar		[25]
Units		NA
Range		Active or Inactive
Default		Inactive

APCDisableRefSourceMarker		
`Status% = APCDisableRefSourceMarker()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command disables the reference source marker from resetting the reference position:  When a marker occurs, the Source Reference Marker Position [23], and the Integrated Source Reference Marker Position [30] may be used to display a marker captured position; this operates in a similar to the way the freeze function. For a marker pulse position capture to occur, the reference marker flag must be enabled by using the command *APCEnableReferenceMarker()*, and the reference marker flag must be reset by using the command *APCResetRefSourceMarkerFlag()* [24] = 0. Read parameter [24] will change state from 0 to 1 when a marker pulse has occurred
APCReadPar		[25]
Units		NA
Range		Active or Inactive
Default		Active

APCEnableFbckSourceMarker		
`Status% = APCEnableFbckSourceMarker()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command enables the feedback source marker to reset the feedback position within one revolution, so that zero position and the marker pulse are aligned:.  For this to happen, the feedback marker flag must be enabled by using the command *APCEnableFeedbackMarker()*, and the feedback marker flag must be reset by using the command *APCResetFbckSourceMarkerFlag()* [44]. If the motor has to turn less than 180° before the marker is found, then the feedback position counters will jump down to zero i.e. zero position turn 0. If the motor has to turn more than 180° before the marker is found, then the feedback position counters will jump up to zero i.e. zero position, turn 1. The turns count is not reset to zero after a marker reset. Once the position counters are aligned this function will have no effect. A Feedback Marker Position capture is not possible when this command is active.
APCReadPar		[45]
Units		NA
Range		Active or Inactive
Default		Inactive

APCDisableFbckSourceMarker		
`Status% = APCDisableFbckSourceMarker()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command disables the feedback source marker from resetting the feedback position:  When a marker occurs, the Source Feedback Marker Position [43], and the Integrated Source Feedback Marker Position [50] may be used to display a marker captured position; this operates in a similar to the way the freeze function. For a marker pulse position capture to occur, the feedback marker flag must be enabled by using the command *APCEnableFeedbackMarker()*, and the feedback marker flag must be reset by using the command *APCResetFbckSourceMarkerFlag()* [44] = 0. Read parameter [44] will change state from 0 to 1 when a marker pulse has occurred
APCReadPar		[45]
Units		NA
Range		Active or Inactive
Default		Active

APCResetRefSourceMarkerFlag		
`Status% = APCResetRefSourceMarkerFlag()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
	-3:	Another process is modifying the object's data
Description		This command resets the Reference marker capture flags: - - APC reference marker flag [24]. - The virtual parameter reference marker flag #90.41. - The selected source drive marker parameter*. * Drive encoder marker flag (Pr3.32) * Option module marker flag (x.08). For a marker pulse position capture to occur, the Reference marker flag must be enabled by using the command *APCEnableReferenceMarker()*.
APCReadPar		[24]
Units		NA
Range		Active or Inactive
Default		Inactive

APCResetFbckSourceMarkerFlag		
`Status% = APCResetFbckSourceMarkerFlag()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
	-3:	Another process is modifying the object's data
Description		This command resets the feedback marker capture flags: - - APC feedback marker flag [44]. - The virtual parameter feedback marker flag #90.42. - The selected source drive marker parameter*. * Drive encoder marker flag (Pr3.32) * Option module marker flag (x.08). For a marker pulse position capture to occur, the feedback marker flag must be enabled by using the command *APCEnableFeedbackMarker()*.
APCReadPar		[44]
Units		NA
Range		Active or Inactive
Default		Inactive

APCResetRefSourceFreezeFlag		
`Status% = APCResetRefSourceFreezeFlag()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
	-3:	Another process is modifying the object's data
Description		This command resets the Reference Freeze capture flags: - - APC reference marker flag [22]. - The virtual parameter feedback freeze flag #90.28. - The selected source drive marker parameter*. * Option module Freeze flag (x.39). For a Freeze position capture to occur, the Reference freeze flag must be enabled by using the command *APCEnableReferenceFreeze()*.
APCReadPar		[22]
Units		NA
Range		Active or Inactive
Default		Inactive

APCResetFbckSourceFreezeFlag		
`Status% = APCResetFbckSourceFreezeFlag()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
	-3:	Another process is modifying the object's data
Description		This command resets the Reference Freeze capture flags: - - APC feedback marker flag [42] - The virtual parameter feedback freeze flag #90.18. - The selected source drive marker parameter*. * Option module Freeze flag (x.39). For a Freeze position capture to occur, the Feedback freeze flag must be enabled by using the command *APCEnableFeedbackFreeze()*.
APCReadPar		[42]
Units		NA
Range		Active or Inactive
Default		Inactive

APCEnableReferenceMarker		
`Status% = APCEnableReferenceMarker()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
Description		This command enables the reference encoder marker pulse to be seen by the SM-Applications / APC. This command is the equivalent of setting #90.45 = 1. This command must be active for a reference marker position reset, or a reference marker position capture to be performed. This command is made inactive by *APCDisableReferenceMarker().*
APCReadPar		No read parameter; use #90.45 instead
Units		NA
Range		Active or Inactive
Default		Inactive

APCDisableReferenceMarker		
`Status% = APCDisableReferenceMarker()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
Description		This command disables the reference encoder marker pulse from being seen by the SM-Applications / APC. This command is the equivalent of setting #90.45 = 0. This command is made inactive by *APCEnableReferenceMarker().*
APCReadPar		No read parameter; use #90.45 instead
Units		NA
Range		Active or Inactive
Default		Active

APCEnableFeedbackMarker		
`Status% = APCEnableFeedbackMarker()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
Description		This command enables the feedback encoder marker pulse to be seen by the SM-Applications / APC. This command is the equivalent of setting #90.46 = 1. This command must be active for a feedback marker position reset, or a feedback marker position capture to be performed. This command is made inactive by *APCDisableFeedbackMarker().*
APCReadPar		No read parameter; use #90.46 instead
Units		NA
Range		Active or Inactive
Default		Inactive

APCDisableFeedbackMarker		
`Status% = APCDisableFeedbackMarker()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
Description		This command disables the feedback encoder marker pulse from being seen by the SM-Applications / APC. This command is the equivalent of setting #90.46 = 0. This command is made inactive by *APCEnableFeedbackMarker().*
APCReadPar		No read parameter; use #90.46 instead
Units		NA
Range		Active or Inactive
Default		Active

APCEnableReferenceFreeze		
`Status% = APCEnableReferenceFreeze()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
Description		This command enables the reference encoder freeze position data to be seen by the SM-Applications / APC. This command is the equivalent of setting #90.47 = 1. This command must be active for a reference freeze position capture to be performed. This command is made inactive by *APCDisableReferenceFreeze().*
APCReadPar		No read parameter; use #90.47 instead
Units		NA
Range		Active or Inactive
Default		Inactive

APCDisableReferenceFreeze		
`Status% = APCDisableReferenceFreeze()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
Description		This command disables the reference encoder freeze position data from being seen by the SM-Applications / APC. This command is the equivalent of setting #90.47 = 0. This command is made inactive by *APCEnableReferenceFreeze().*
APCReadPar		No read parameter; use #90.47 instead
Units		NA
Range		Active or Inactive
Default		Active

APCEnableFeedbackFreeze		
`Status% = APCEnableFeedbackFreeze()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
Description		This command enables the feedback encoder freeze position data to be seen by the SM-Applications / APC. This command is the equivalent of setting #90.48 = 1. This command must be active for a feedback freeze position capture to be performed. This command is made inactive by *APCDisableFeedbackFreeze().*
APCReadPar		No read parameter; use #90.48 instead
Units		NA
Range		Active or Inactive
Default		Inactive

APCDisableFeedbackFreeze		
`Status% = APCDisableFeedbackFreeze()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
Description		This command disables the feedback encoder freeze position data from being seen by the SM-Applications / APC. This command is the equivalent of setting #90.48 = 0. This command is made inactive by *APCEnableFeedbackFreeze().*
APCReadPar		No read parameter; use #90.48 instead
Units		NA
Range		Active or Inactive
Default		Active

APCInvertRefSource		
`Status% = APCInvertRefSource()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command inverts the reference source position, after the source counter [20], but before the integrated counter [28]. This command is active when the source reference invert flag [27] = 1.
APCReadPar		[27]
Units		NA
Range		Active or Inactive
Default		Inactive

APCDoNotInvertRefSource		
`Status% = APCDoNotInvertRefSource()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command cancels a reference source position invert (*APCInvertRefSource*, [27] = 1), and makes the reference non-inverted. This command is active when the source reference invert flag [27] = 0.
APCReadPar		[27]
Units		NA
Range		Active or Inactive
Default		Active

APCInvertFbckSource		
`Status% = APCInvertFbckSource()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command inverts the feedback source position, after the source counter [40], but before the integrated counter [48]. This command is active when the source feedback invert flag [47] is 1.
APCReadPar		[47]
Units		NA
Range		Active or Inactive
Default		Inactive

APCDoNotInvertFbckSource		
`Status% = APCDoNotInvertFbckSource()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command cancels a feedback source position invert (*APCInvertFbckSource*, [47] = 1), and makes the feedback non-inverted. This command is active when the source feedback invert flag [47] = 0.
APCReadPar		[47]
Units		NA
Range		Active or Inactive
Default		Active

APCSetReferencePosition		
`Status% = APCSetReferencePosition(Turns%, Position%, PositionFine%)`		
Input Arguments	*Turns%*:	Signed 32-bit. The lower 16-bits of Turns% will be treated as a signed 16-bit value. Bits 16 to 31 of Turns% will be ignored
	Position%:	Signed 32-bit. The lower 16-bits of Position% will be treated as a signed 16-bit value. Bits 16 to 31 of Position% will be ignored
	PositionFine%:	Signed 32-bit. The lower 16-bits of PositionFine% will be treated as a signed 16-bit value. Bits 16 to 31 of PositionFine% will be ignored
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
	-3:	Another process is modifying the object's data
Description		This command sets the user reference source position. This user reference position is used when reference source selector is set to 4, set by the command *APCSetReferenceSource(4)*. **This command must only be used in the POS Task.**
APCReadPar		[20] and [28]
Units		Turns and Encoder Counts
Range		16bit Turns, 16bit coarse Position, 16bit Fine Position
Default		0

APCSetFeedbackPosition		
`Status% = APCSetFeedbackPosition(Turns%, Position%, PositionFine%)`		
Input Arguments	*Turns%*:	Signed 32-bit. The lower 16-bits of Turns% will be treated as a signed 16-bit value. Bits 16 to 31 of Turns% will be ignored
	Position%:	Signed 32-bit. The lower 16-bits of Position% will be treated as a signed 16-bit value. Bits 16 to 31 of Position% will be ignored
	PositionFine%:	Signed 32-bit. The lower 16-bits of PositionFine% will be treated as a signed 16-bit value. Bits 16 to 31 of PositionFine% will be ignored
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
	-3:	Another process is modifying the object's data
Description		This command sets the user reference source position. This user reference position is used when reference source selector is set to 4, set by the command *APCSetFeedbackSource(4)*. **This command must only be used in the POS Task.**
APCReadPar		[40] and [48]
Units		Turns and Encoder Counts
Range		16bit Turns, 16bit coarse Position, 16bit Fine Position
Default		0

APCGetReferenceStatus		
`(Source%, Status%) = APCGetReferenceStatus()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
	-3:	Another object is modifying the object's data
	Source%:	
	0:	Drive
	1:	Slot 1
	2:	Slot 2
	3:	Slot 3
	4:	User program
	5:	No source selected
Description		This command retrieves the current selected reference source, set by the command *APCSetReferenceSource(Source%)* and will put the source number into a user variable; in this case `Source%`.
APCReadPar		NA
Units		NA
Range		0 to 5
Default		0

APCGetFeedbackStatus		
`(Source%, Status%) = APCGetFeedbackStatus()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
	-3:	Another object is modifying the object's data
	Source%:	
	0:	Drive
	1:	Slot 1
	2:	Slot 2
	3:	Slot 3
	4:	User program
	5:	No source selected
Description		This command retrieves the current selected feedback source, set by the command *APCSetFeedbackSource(Source%)* and will put the source number into a user variable; in this case `Source%`.
APCReadPar		NA
Units		NA
Range		0 to 5
Default		0

APCEnableRefInput		
`Status% = APCEnableRefInput()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
Description		This command enables the reference source position data to be passed through to the Reference Input [31]. This command is active when the reference input disable flag [32] = 0. This command is made inactive by *APCDisableRefInput().*
APCReadPar		[32]
Units		NA
Range		Active or Inactive
Default		Active

APCDisableRefInput		
`Status% = APCDisableRefInput()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
Description		This command disables the reference source position data from being passed through to the Reference Input [31]. This allows source reference position [20], to be manipulated e.g. a filter. The resultant manipulated position data may then be passed to the reference input using the command *APCSetRefInput(RefIn%)* This command is active when the reference input disable flag [32] = 1. This command is made inactive by *APCEnableRefInput().*
APCReadPar		[32]
Units		NA
Range		Active or Inactive
Default		Inactive

APCEnableFbckInput		
`Status% = APCEnableFbckInput()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
Description		This command enables the feedback source position data to be passed through to the Feedback Input [51]. This command is active when the feedback input disable flag [52] = 0. This command is made inactive by *APCDisableFbckInput().*
APCReadPar		[52]
Units		NA
Range		Active or Inactive
Default		Active

APCDisableFbckInput		
`Status% = APCDisableFbckInput()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
Description		This command disables the feedback source position data from being passed through to the Feedback Input [51]. This allows source feedback position [40], to be manipulated e.g. a filter. The resultant manipulated position data may then be passed to the reference input using the command *APCSetFbckInput(FbckIn%)* This command is active when the feedback input disable flag [52] = 1. This command is made inactive by *APCEnableFbckInput()*.
APCReadPar		[52]
Units		NA
Range		Active or Inactive
Default		Inactive

APCSetRefInput		
`Status% = APCSetRefInput(RefIn%)`		
Input Arguments	*RefIn%*	Signed 32bit position in encoder counts only
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
Description		This command sets the reference source input position data [31], when the command *APCDisableRefInput()* is active [32] = 1. Position data entered in POS0 using this command is used in the same POS task cycle
APCReadPar		[31]
Units		NA
Range		Active or Inactive
Default		0

APCSetFbckInput		
`Status% = APCSetFbckInput(FbckIn%)`		
Input Arguments	*FbckIn%*	Signed 32bit position in encoder counts only
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful
Description		This command sets the reference source input position data [51], when the command *APCDisableFbckInput()* is active [52] = 1. Position data entered in POS0 using this command is used in the same POS task cycle
APCReadPar		[51]
Units		NA
Range		Active or Inactive
Default		0

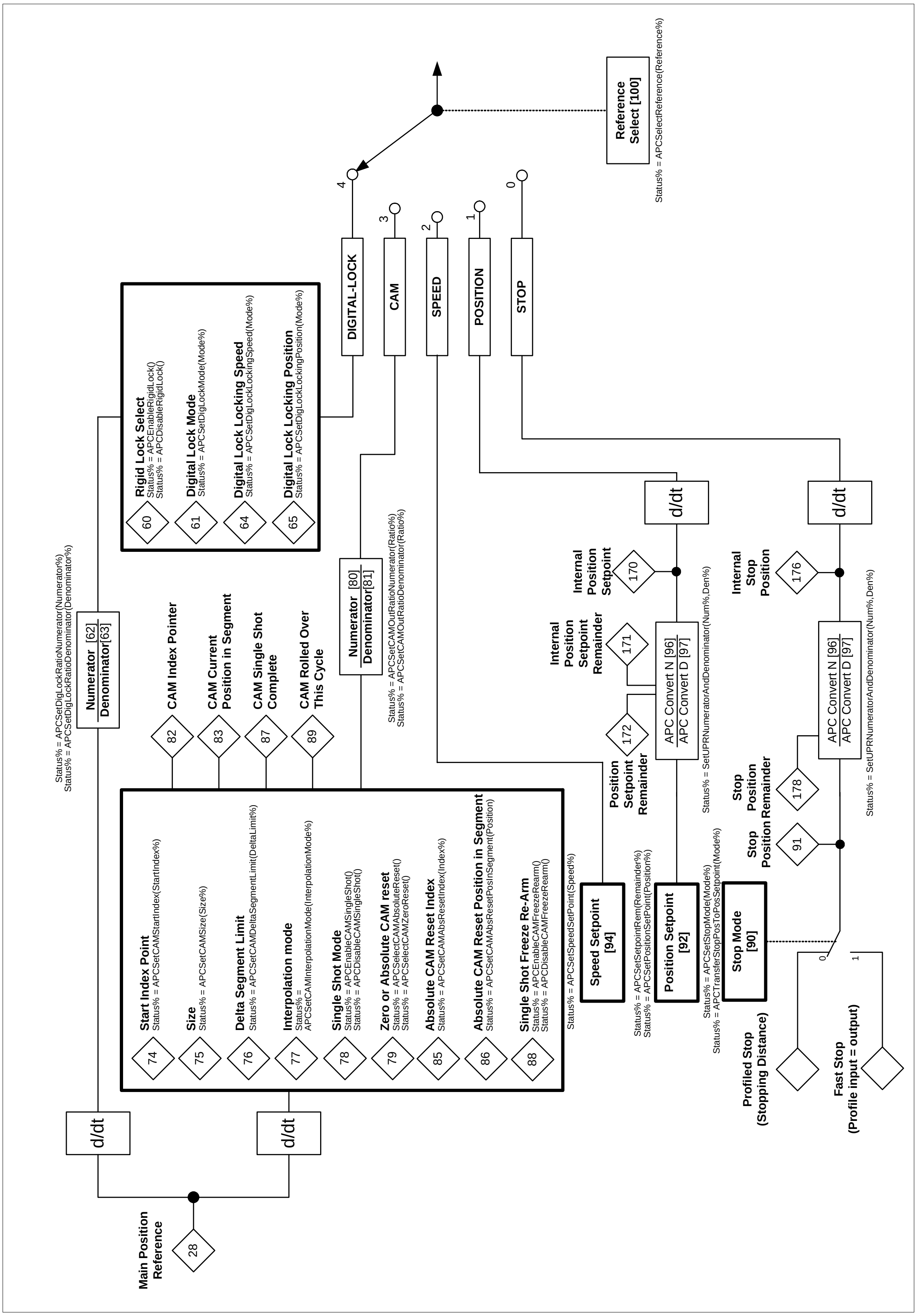
5.2.3 CTSync Functions

CTSyncSetMasterReferences		
`CTSyncSetMasterReferences(Reference1%, Reference2% AuxRef%)`		
Input Arguments	*Reference1%:*	Signed 32bit value
	Reference2%:	Signed 32bit value
	AuxRef%:	Unsigned 8bit value
Description		This function block allows the CTSync Master to write reference data to all CTSync Slaves on the network. It may only be used on the Master.
Units		NA
Range	Reference1%: Reference2%: AuxRef%:	-2^{31} to 2^{31} - 1 -2^{31} to 2^{31} - 1 0 to 2^{8}-1
Default		0

CTSyncGetSlaveReferences		
`(Reference1%, Reference2% AuxRef%, Status%) = CTSyncGetSlaveReferences()`		
Output Arguments	*Reference1%:*	Signed 32bit value
	Reference2%:	Signed 32bit value
	AuxRef%:	Unsigned 8bit value
	Status%:	
	1:	Operation successful
	0	Zero or fewer bytes than expected received
	-1	More bytes than expected received
	-2	Checksum error in received data
Description		This function block allows the CTSync Slaves to read the reference data created by the Master. It can be used on both the Master and Slaves. The block returns the values of the last Master references received. The function block returns the values of the last Master references received.Profile Generator.
Units		NA
Range		NA
Default		0

5.2.4 References

Figure 5-4



5.2.4.1 Reference Select

APCSelectReference		
`Status% = APCSelectReference(Reference%)`		
Input Arguments	*Reference%*:	
	0:	Use Stop position as reference
	1:	Use Position setpoint as reference
	2:	Use Speed setpoint as reference
	3:	Use CAM as reference
	4:	Use Digital Lock as reference
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description	*Reference%*	
	0:	With `Reference%` set to 0, the Stop reference will be enabled [100] = 0. The type of stopping mode is selected by the command *APCSetStopMode*(Mode%) [90]. See the *Stop Reference* section for functional information on this reference.
	1:	With `Reference%` set to 1, the Position reference will be enabled [100] = 1. The Position reference setpoint is set by the command *APCSetPositionSetPoint(Position%)* [92]. See the *Position Reference* section for functional information on this reference.
	2:	With `Reference%` set to 2, the Speed reference will be enabled [100] = 2. The Speed Setpoint is set by the command *APCSetSpeedSetPoint(Speed%)* [94]. See the *Speed Reference* section for functional information on this reference.
	3:	With `Reference%` set to 3, the CAM Function Generator will be enabled [100] = 3. See the *CAM Reference* section for functional information on this reference.
	4:	With `Reference%` set to 4, the Digital Lock reference will be enabled [100] = 4. See the *Digital Lock Reference* section for functional information on this reference.
APCReadPar		[100]
Units		NA
Range		0 to 4
Default		0

APCSelectActionOnFreeze		
`Status% = APCSelectActionOnFreeze(Action%)`		
Input Arguments	*Action%*:	
	0:	No action on freeze event
	1:	Enable digital lock on freeze event
	2:	Enable CAM on freeze event
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description	*Action%*:	
	0:	With `Action%` set to 0, there will be no additional action when a Freeze event occurs.
	1:	With `Action%` set to 1, the Digital Lock reference will be selected [100] = 4, when a freeze event occurs. To allow this function to happen the Freeze function must be enabled by using the command *APCEnableReferenceFreeze()*, and the Freeze flag must be reset, using the command *APCResetRefSourceFreezeFlag()*.
	2:	With `Action%` set to 2, the CAM reference will be selected [100] = 3, when a freeze event occurs. To allow this function to happen the Freeze function must be enabled by using the command *APCEnableReferenceFreeze()*, and the Freeze flag must be reset, using the command *APCResetRefSourceFreezeFlag().*
APCReadPar		[100]
Units		NA
Range		0 to 2
Default		0

5.2.4.2 Stop Reference

APCSetStopMode		
`Status% = APCSetStopMode(Mode%)`		
Input Arguments	*Mode%*:	
	0:	Stop, using the currently selected profile
	1:	Stop, without using the profile
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description	*Mode%*:	
	0:	In this mode, when the Stop reference is set, using the command *APCSelectReference(0)* [100] = 0, the profile generator will calculate a stop position which can be achieved using the profile deceleration rate, set by the command, *APCSetProfileDecelRate(Decel%)* [111]. The calculated stop position is shown by [91]. In SM-Apps Firmware =>V01.04.04 the stop position shown by [91] may be read in user units by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
	1:	In this mode, when the Stop reference is set, using the command *APCSelectReference(0)* [100] = 0, the profile generator is bypassed, and the feedback will stop instantly with no ramps. Care should be taken when using this mode as an instant stop could damage the application mechanics.
APCReadPar		[90] and [91]
Units		NA
Range		0 or 1APCTransferStopPosToPosSetPoint
Default		0

APCTransferStopPosToPosSetPoint		
`Status% = APCTransferStopPosToPosSetPoint()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		When this command is called, the stop position [91] and remainder [178] if applicable, is transferred to the Position setpoint [92] and remainder [172]. When the stop reference is not selected, [91] is continually updated with the final resting position of the axis if a stop were called in that particular sample, with the current main profile generator settings. This command is useful when running the APC in user units, so that when transferring from the Stop Reference to the Position reference, any positional remainder is also transferred, and therefore no unexpected motion e.g. if the stop position is 111 units and 2345 remainder, if only the units were transferred, the axis would rotate forwards by the equivalent position to the remainder of 2345.
APCReadPar		[91] and [178]
Units		NA
Range		NA
Default		NA

5.2.4.3 Position Reference

APCSetPositionSetPoint		
`Status% = APCSetPositionSetPoint(Position%)`		
Input Arguments	*Position%*:	Signed 32bit position
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the Position setpoint [92], and will be applied to the profile generator input when selected, using the command *APCSelectReference(1)* [100] = 1. The value of `Position%` is entered in counts, and should be calculated with respect to the same resolution set for the feedback encoder, defined by the command *APCSetNumOfTurnsBits(TurnBits%)* [14]. In SM-Apps Firmware =>V01.04.04 the position setpoint may be set in user units with remainder, by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[92]
Units		Encoder Counts or User Units
Range		-2^{31} to $2^{31} - 1$
Default		0

APCSetSetPointRem		
`Status% = APCSetSetPointRem(Remainder%)`		
Input Arguments	*Remainder%*:	Signed 32bit position remainder
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the Position setpoint remainder [172], and will be applied to the profile generator input when selected, using the command *APCSelectReference(1)* [100] = 1. This command is only available in SM-Apps Firmware =>V01.04.04, where the position setpoint may be set in user units with remainder by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[92]
Units		Encoder Counts or User Units
Range		-2^{31} to $2^{31} - 1$
Default		0

5.2.4.4 Speed Reference

APCSetSpeedSetPoint		
`Status% = APCSetSpeedSetPoint(Speed%)`		
Input Arguments	*Speed%*:	Signed 32-bit position
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the Speed setpoint [94], and will be applied to the profile generator input when selected, using the command *APCSelectReference(2)* [100] = 2. In SM-Apps Firmware =>V01.04.04 the speed setpoint may be set in user units/s by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[94]
Units		2^{32} counts / rev encoder over 250μs or User units/s
Range		-715827883 to 715827883
Default		0

5.2.4.5 CAM Reference

APCCamInitialise		
`Status% = APCCamInitialise(InArray%, OutArray1%, OutArray2%, InterpolationArray%)`		
Input Arguments	*InArray%*:	Input array dimensioned by the user
	OutArray1%:	Output array 1 dimensioned by the user
	OutArray2%:	Output array 2 dimensioned by the user
	InterpolationArray%:	Interpolation array dimensioned by the user
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command defines the names of the arrays to be used with the APC CAM reference. With reference to the above, the first array, `InArray%`, is the name of the array which defines the reference (x axis) position elements for each CAM segment. The second array, `OutArray1%`, is the name of the array which defines the feedback (y axis) position elements for each CAM segment.The interpolation used with respect to the reference array `InArray%`, is defined by *APCSetCAMInterpolationMode(n).* The third array, `OutArray2%`, is the name of the array which defines the feedback offset position elements / CAM segments, the value of this array is added to the `OutArray1%` array, but it only uses linear interpolation with respect to the reference array `InArray%`. The fourth array, `InterpolationArray%`, is the name of the array which defines the interpolation type used for `OutArray1%` with respect to the reference array, `InArray%` for each CAM segment. This is only applicable when multiple interpolation is required, defined by *APCSetCAMInterpolationMode(-1).* The `InArray%` and the `OutArray1%` are both required for the basic operation of the CAM reference, however where the `OutArray2%` array, and the Interpolation array are not required, then use: - *APCCamInitialise1* *APCCamInitialise2* *APCCamInitialise3*
APCReadPar		NA
Units		NA
Range		NA
Default		NA

APCCamInitialise1		
`Status% = APCCamInitialise1(InArray%, OutArray1%)`		
Input Arguments	*InArray%*:	Input array dimensioned by the user
	OutArray1%:	Output array 1 dimensioned by the user
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command defines the names of the arrays to be used with the APC CAM reference. With reference to the above, the first array, `InArray%`, is the name of the array which defines the reference (x axis) position elements for each CAM segment. The second array, `OutArray1%`, is the name of the array which defines the feedback (y axis) position elements for each CAM segment. The interpolation used with respect to the reference array `InArray%`, is defined by *APCSetCAMInterpolationMode(n).* When this CAM Initialisation command is used, the Interpolation type must be defined by the command *APCSetCAMInterpolationMode(0)*, or *(1)*, which means that all moves will be interpolated using Linear [77] = 0, or Cosine interpolation [77] = 1.
APCReadPar		NA
Units		NA
Range		NA
Default		NA

APCCamInitialise2		
`Status% = APCCamInitialise2(InArray%, OutArray1%, InterpolationArray%)`		
Input Arguments	*InArray%*:	Input array dimensioned by the user
	OutArray1%:	Output array 1 dimensioned by the user
	InterpolationArray%	Interpolation array dimensioned by the user
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command defines the names of the arrays to be used with the APC CAM reference. With reference to the above, the first array, `InArray%`, is the name of the array which defines the reference (x axis) position elements for each CAM segment. The second array, `OutArray1%`, is the name of the array which defines the feedback (y axis) position elements for each CAM segment. The interpolation used with respect to the reference array `InArray%`, is defined by *APCSetCAMInterpolationMode(n).* The third array, `InterpolationArray%`, is the name of the array which defines the interpolation type used for `OutArray1%` with respect to the reference array, `InArray%`, for each CAM segment. This is only applicable when multiple interpolation is required, defined by *APCSetCAMInterpolationMode(-1).*
APCReadPar		NA
Units		NA
Range		NA
Default		NA

APCCamInitialise3		
`Status% = APCCamInitialise3(InArray%, OutArray1%, OutArray2%)`		
Input Arguments	*InArray%*:	Input array dimensioned by the user
	OutArray1%:	Output array 1 dimensioned by the user
	OutArray2%	Output 2 array dimensioned by the user
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command defines the names of the arrays to be used with the APC CAM reference. With reference to the above, the first array, `InArray%`, is the name of the array which defines the reference (x axis) position elements for each CAM segment. The second array, `OutArray1%`, is the name of the array which defines the feedback (y axis) position elements for each CAM segment.The interpolation used with respect to the reference array `InArray%`, is defined by *APCSetCAMInterpolationMode(n).* The third array, `OutArray2%`, is the name of the array which defines the feedback offset position elements / CAM segments, the value of this array is added to the `OutArray1%` array, but it only uses linear interpolation with respect to the reference array `InArray%`. When this CAM Initialisation command is used the Interpolation type must be defined by the command *APCSetCAMInterpolationMode(0)*, or *(1)*, which means that all moves will be interpolated using Linear [77] = 0, or Cosine interpolation [77] = 1.
APCReadPar		NA
Units		NA
Range		NA
Default		NA

APCSetCAMStartIndex		
`Status% = APCSetCAMStartIndex(StartIndex%)`		
Input Arguments	*StartIndex%*:	Unsigned 16bit. Negative values will be clamped to zero, values over 65535 will be clamped to 65535
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command defines the CAM array start element. If the start index and size are set along with selecting the CAM reference, then the new CAM size and start index will not take effect until the CAM reaches the end and wraps over. To prevent this effect the user should set the CAM size and start index, then in the next Pos task sample select the CAM Reference. See *APCSetCAMSize*.
APCReadPar		[74]
Units		NA
Range		0 to 65535
Default		0

APCSetCAMSize		
`Status% = APCSetCAMSize(Size%)`		
Input Arguments	*Size%*:	Unsigned 16bit. Negative values will be clamped to zero, values over 65535 will be clamped to 65535
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This defines the array size from the array start element, the CAM will use. e.g. Where the arrays contain 20 elements each, and the CAM start index is set to 5, *APCSetCAMStartIndex (5)*, and the CAM size is set to 10, only array elements / CAM segments 5 to 15 will be used. If the CAM size is set to 0, then the Stop reference will be automatically selected, as the CAM reference array requires a minimum of 1 element. If the start index and size are set along with selecting the CAM reference, then the new CAM size and start index will not take effect until the CAM reaches the end and wraps over. To prevent this effect the user should set the CAM size and start index, then in the next Pos task sample select the CAM Reference.
APCReadPar		[75]
Units		NA
Range		0 to 65535
Default		0

APCSetCAMDeltaSegLimit		
`Status% = APCSetCAMDeltaSegLimit(DeltaLimit%)`		
Input Arguments	*DeltaLimit%*:	Unsigned 16bit. Negative values will be clamped to zero, values over 65535 will be clamped to 65535
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the maximum delta position change per POS task clock time. The POS task clock time is set by #81.12 or X.12 for the SM.applications module. If this limit is exceeded, then the reference will automatically be switched to Stop and trip Tr81 on the relevant Unidrive-SP slot.
APCReadPar		[76]
Units		Encoder counts / POS task clock time
Range		0 to 65535
Default		256

APCSetCAMInterpolationMode		
`Status% = APCSetCAMInterpolationMode(InterpolationMode%)`		
Input Arguments	*InterpolationMode%*:	
	0:	All segments use linear interpolation
	1:	All segments use cosine interpolation
	-1:	Interpolation defined by the Interpolation array
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description	*InterpolationMode%*:	
	-1:	This interpolation mode, [77] = -1, allows the interpolation to be defined for each CAM segment of the output array 1, with respect to the input array. The interpolation array used is named in the one of the following commands: - *APCCamInitialise* *APCCamInitialise2* There are 6 different types of interpolation available, which is need to be defined within the interpolation array: 0 = Linear 1= Cosine 2 = Square1 3 = Square2 4 = Cosine1 5 = Cosine2 These values should be set in the corresponding array element. Refer to the *Interpolation* section for more detail of interpolation types.
	0:	With this interpolation mode, [77] = 0, only linear interpolation is between all CAM segment positions.
	1:	With this interpolation mode, [77] = 1, only cosine interpolation is between all CAM segment positions.
APCReadPar		[77]
Units		NA
Range		-1 to 1
Default		0

APCSetCAMOutRatioNumerator		
`Status% = APCSetCAMOutRatioNumerator(Ratio%)`		
Input Arguments	*Ratio%*:	Unsigned 31bit value to set as numerator. Negative values will be clamped at zero.
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the CAM output numerator [80]. The CAM output will be multiplied by the numerator value set, before the profile generator input. It is used in conjunction with the CAM output denominator *APCSetCAMOutRatioDenominator(Denominator%).*
APCReadPar		[80]
Units		NA
Range		0 to 2147483647
Default		1000

APCSetCAMOutRatioDenominator		
`Status% = APCSetCAMOutRatioDenominator(Ratio%)`		
Input Arguments	*Ratio%*:	Unsigned 31bit value to set as denominator. Negative values will be clamped at zero.
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the CAM output denominator [81]. The CAM output will be divided by the denominator value set, before the profile generator input. It is used in conjunction with the CAM output numerator *APCSetCAMOutRatioNumerator(Numerator%).*
APCReadPar		[81]
Units		NA
Range		1 to 2147483647
Default		1000

APCSelectCAMAbsoluteReset		
`Status% = APCSelectCAMAbsoluteReset()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		When this command is active, [79] = 1, the start index and segment position within the defined CAM output array/s can be set by the following commands:- *APCSetCAMAbsResetIndex(Index%)* [82] *APCSetCAMAbsResetPositionInSeg(Position%)* [83]. This can only be achieved when the CAM reference is not selected, w*APCSelectReference(0)*, *(1)*, *(2)*, *(4)*, [100] = 0, 1, 2, 4, or the APC is in disable mode, *APCSetRunMode(1)*, [0] = 0. This command disables Zero CAM reset mode, *APCSelectCAMZeroReset()*
APCReadPar		[79]
Units		NA
Range		Active or Inactive
Default		Inactive

APCSelectCAMZeroReset		
`Status% = APCSelectCAMZeroReset()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		When this command is active, [79] = 0, the start index position within the defined CAM is set by the command *APCSetCAMStartIndex(StartIndex%)*, the position within the index segment is set to zero. This command disables Absolute CAM reset mode *APCSelectCAMAbsoluteReset()*.
APCReadPar		[79]
Units		NA
Range		Active or Inactive
Default		Active

APCSetCAMAbsResetIndex		
`Status% = APCSetCAMAbsResetIndex(Index%)`		
Input Arguments	Index%	The number of the CAM segment to be reset to. 0 to 65535 inclusive.
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the value of the CAM absolute reset index, when the command *APCSelectCAMAbsoluteReset()* is active. The value entered into `Index%` is used to define where within the defined CAM arrays, the CAM reference will start from, when it is next selected. The index can only be set when the CAM reference is not selected, *APCSelectReference(0)*, *(1)*, *(2)*, *(4)*, [100] = 0, 1, 2, 4, or the APC is in disable mode, *APCSetRunMode(1)* [1] = 0. The Index value can be read from read parameter [82].
APCReadPar		[85]
Units		NA
Range		0 to 65535
Default		0

APCSetCAMAbsResetPositionInSeg		
`Status% = APCSetCAMAbsResetPositionInSeg(Position%)`		
Input Arguments	Position%	0 to (2^{31} - 1) position
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the value of the CAM absolute reset position in index segment when the command *APCSelectCAMAbsoluteReset()* is active. The value entered into `Position%` is used to define where within the index segment, defined by *APCSetCAMAbsResetIndex(Index%)*, the CAM reference will start from, when it is next selected. The position value can only be set when the CAM reference is not selected, *APCSelectReference(0)*, *(1)*, *(2)*, *(4)*, [100] = 0, 1, 2, 4, or the APC is in disable mode, *APCSetRunMode(1)* [1] = 0. The current CAM position value can be read from read parameter [83].
APCReadPar		[86]
Units		Encoder Counts
Range		0 to 2^{31} - 1
Default		0

APCEnableCAMSingleShot		
`Status% = APCEnableCAMSingleShot()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command enables CAM reference single shot mode. When the CAM reference is selected it will run one cycle of the CAM in either forward or reverse direction, depending on the direction of the reference encoder, and then stop. To re-start the CAM once a single shot cycle has finished:- - De-selected then re-selected the CAM reference - Call an APC reset using the command *APCReset()* - Disable then re-enable Single shot mode; disabled by *APCDisableCAMSingleShot()* - Toggle the APC run mode from 2 to any other run mode. When *APCSelectCAMZeroReset()* is active: The CAM will start at the first CAM segment defined by the command *APCSetCAMStartIndex(StartIndex%)*, and run the CAM in either the forward or reverse direction until the CAM start index is next reached, at which point the CAM will stop. When *APCSelectCAMAbsoluteReset()* is active: The CAM will Start from the absolute CAM reset index, and Position in segment, defined by the commands *APCSetCAMAbsResetIndex(Index%)*, and *APCSetCAMAbsResetPositionInSeg(Position%)*. When the CAM is started it will only be able to complete the remaining CAM points in either direction, until the CAM start index is crossed. e.g. with a CAM of 1 segment, containing 10000 counts, started in single shot at 1000 counts, will be able to move forward by 9000 counts, or reverse by 1000 before the CAM will stop. This command makes *APCDisableCAMSingleShot()* inactive. Read parameter [87] indicates when a single shot cycle has completed.
APCReadPar		[78] and [87]
Units		NA
Range		Active or Inactive
Default		Inactive

APCDisableCAMSingleShot		
`Status% = APCDisableCAMSingleShot()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command enables continuous CAM mode. When *APCSelectCAMZeroReset()* is active: The CAM will start at the first CAM segment defined by the command *APCSetCAMStartIndex(StartIndex%)*, and run the CAM in either the forward or reverse direction, and will then wrap around to the start index segment, and will continually cycle through the CAM. When *APCSelectCAMAbsoluteReset()* is active: The CAM will Start from the absolute CAM reset index, and Position in segment, defined by the commands *APCSetCAMAbsResetIndex(Index%)*, and *APCSetCAMAbsResetPositionInSeg(Position%)*. The CAM will run to the end of the last segment, then wrap around to the start index segment, and will continually cycle through the CAM. This command makes *APCEnableCAMSingleShot()* inactive. Read parameter [89] indicates the cycle in which the CAM wrapped/rolled over to the start index.
APCReadPar		[79] and [89]
Units		NA
Range		Active or Inactive
Default		Active

APCEnableCAMFreezeRearm		
`Status% = APCEnableCAMFreezeRearm()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command enables the single shot CAM freeze re-arm feature. This feature is for basic registration systems implemented with the CAM reference in single shot, where a registration sensor is used to trigger the reference freeze input and thereby start the CAM. When enabled, and a single shot completes, the stop reference will automatically be selected, the reference freeze will be reset, and the freeze trigger system will be reset, ready for a freeze to trigger the CAM again. This feature saves the user 1 Pos task cycle by internally resetting the system. To use this feature the user must: 1/ Select single shot CAM with *APCEnableCAMSingleShot()* before starting the CAM on a freeze for the first cycle, every time the system is run. 2/ Enable the reference freeze with *APCEnableReferenceFreeze()*, in the initial task. 3/ Reset the Freeze flag with *APCResetRefSourceFreezeFlag()* before starting the CAM on a freeze for the first cycle, every time the system is run. 4/ Select action on freeze with *APCSelectActionOnFreeze(2)* before starting the CAM on a freeze for the first cycle, every time the system is run.
APCReadPar		[88]
Units		NA
Range		Active or Inactive
Default		Inactive

APCDisableCAMFreezeRearm		
`Status% = APCDisableCAMFreezeRearm()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command disables the single shot CAM freeze re-arm feature.
APCReadPar		[88]
Units		NA
Range		Active or Inactive
Default		Active

5.2.4.6 Digital Lock Reference

APCSetDigLockMode		
`Status% = APCSetDigLockMode(Mode%)`		
Input Arguments	*Mode%*:	
	0:	Profile generator active, not locked, but seeking lock
	1:	Profile generator inactive and locked
	2:	Profile generator active and system cannot lock
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description	*Mode%*:	
	0:	In this mode, [61] = 0, the profile generator will try to attain a locked status [61] = 1, within constraints of the profile setup parameters set by commands: *APCSetProfileAccelRate(Accel%)* [110], *APCSetProfileDecelRate(Decel%)* [111], non rigid lock only *APCSetProfileMaxSpeedClamp(MaxSpeed%)* [113]. The profile generator will try to attain either ratio speed synchronisation (non-rigid digital lock), or ratio speed and position synchronisation (rigid digital lock), depending on whether the command *APCEnableRigidLock()* [60] = 1, or *APCDisableRigidLock()*(default)[60] = 0 is active. It is possible to manually set the digital lock mode, however once the locking conditions set by *APCSetDigLockLockingSpeed(LockSpeed%)* [64], and *APCSetDigLockLockingPosition(LockPosition%)* [65] are met, the digital lock mode will automatically change to 1[61] = 1.
	1:	In this mode, [61] = 1, the system is locked, the drive axis is directly coupled in position to the ratio of the reference position. Any difference in position and speed between the reference and feedback will re-gained with no-ramps as the profile generator is bypassed. It is possible to manually set the digital lock mode to 1, however care should be taken when doing this, as switching into digital lock mode 1 causes the profile generator to be bypassed, and therefore any differences between the reference and the feedback position, will lead to an instantaneous change in output speed, which could cause damage to the mechanics of an application.
	2:	In this mode, [61] = 2, the profile generator will try to attain a lock, within constraints of the profile setup parameters set by commands: *APCSetProfileAccelRate(Accel%)* [110] *APCSetProfileDecelRate(Decel%)* [111], non rigid lock only *APCSetProfileMaxSpeedClamp(MaxSpeed%)* [113] The APC will not actually reach a locked status [61] = 1, as in this mode the profile generator parameter setup is permanently active, even when the reference and feedback are synchronised. The profile generator will try to attain either speed synchronisation (non-rigid digital lock), or speed and position synchronisation (rigid digital lock), depending on whether the command *APCEnableRigidLock()* [60] = 1, or *APCDisableRigidLock()*(default)[60] = 0 is active. The only way to select Mode 2, is to select it manually by using the command `Status% = APCSetDigLockMode(2).`
APCReadPar		[61]
Units		NA
Range		0 to 2
Default		0

APCSetDigLockRatioNumerator		
`Status% = APCSetDigLockRatioNumerator(Numerator%)`		
Input Arguments	*Numerator%*:	Unsigned 31bit value to set as numerator
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command sets the digital lock numerator [62]. The Reference position will be multiplied by the numerator value set, before the offsets are applied. It is used in conjunction with the digital lock denominator *APCSetDigLockRatioDenominator(Denominator%).*
APCReadPar		[62]
Units		Number
Range		0 to 2147483647
Default		1000

APCSetDigLockRatioDenominator		
`Status% = APCSetDigLockRatioDenominator(Denominator%)`		
Input Arguments	*Denominator%*:	Unsigned 31bit value to set as denominator
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command sets the digital lock denominator [63]. The Reference position will be divided by the denominator value set, before the offsets are applied. It is used in conjunction with the digital lock numerator *APCSetDigLockRatioNumerator(Numerator%)*.
APCReadPar		[63]
Units		Number
Range		1 to 2147483647
Default		1000

APCSetDigLockLockingSpeed		
`Status% = APCSetDigLockLockingSpeed(Speed%)`		
Input Arguments	*Speed%*:	Unsigned 31bit value to define locking speed
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command sets the digital lock locking speed [64], which is used to define when the digital lock reference is locked. The condition for locking is when the modulus of the difference between reference and feedback speed, is less than or equal to digital lock locking speed. In SM-Apps Firmware =>V01.04.04 the Digital Lock locking speed may be set in user units/s by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[64]
Units		2^{32} counts / rev encoder over 250µs or User units/s
Range		0 to 2^{31}-1
Default		17895

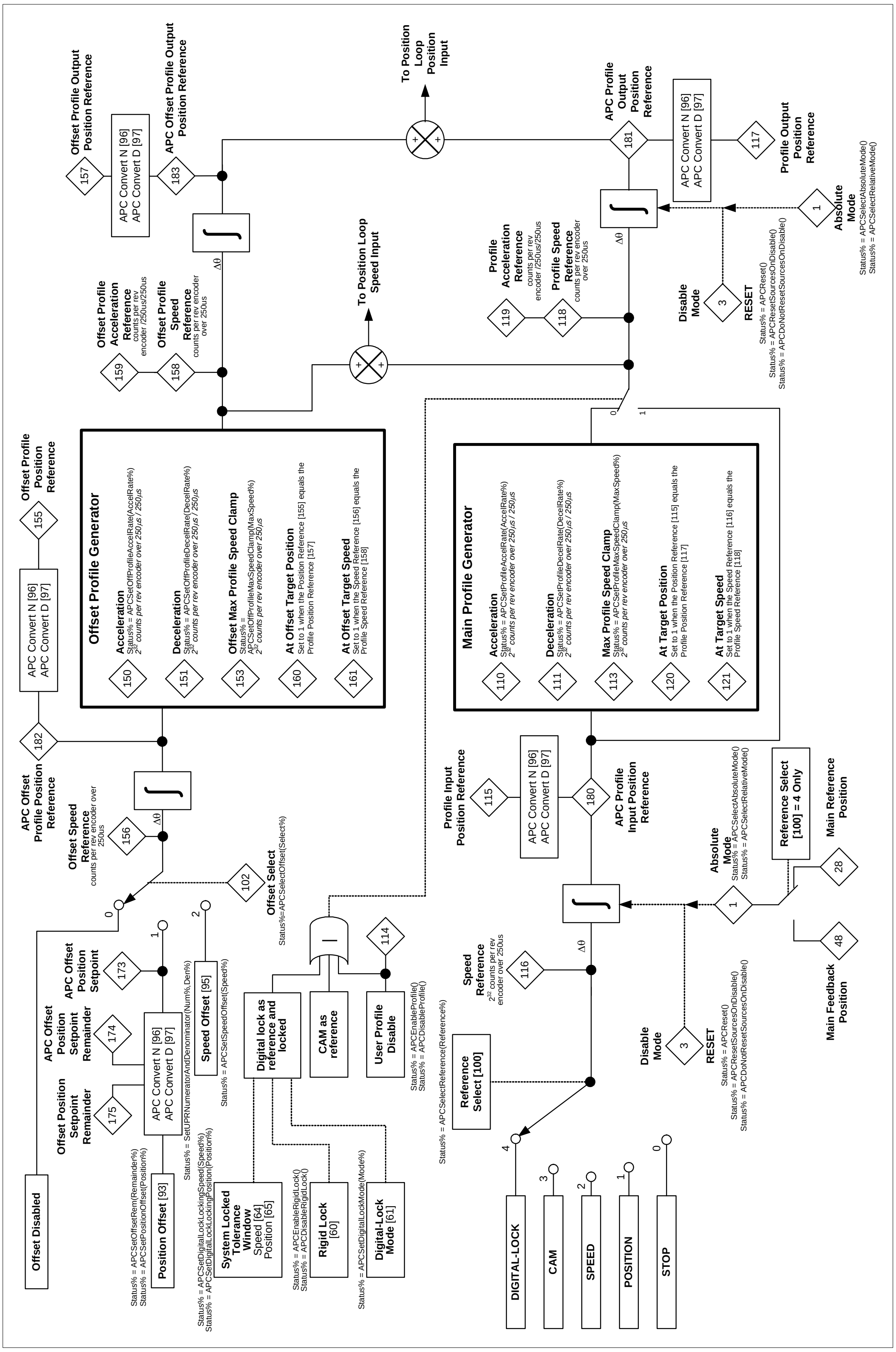
APCSetDigLockLockingPosition		
`Status% = APCSetDigLockLockingPosition(Position%)`		
Input Arguments	*Position%*:	Unsigned 31bit value to define locking position
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command sets the digital lock locking position [65], which is used to define when the digital lock reference is locked, when the command *APCEnableRigidLock()* is active ([60] = 1). The condition for locking is when the modulus of the difference between reference and feedback position, is less than or equal to digital lock locking position. In SM-Apps Firmware =>V01.04.04 the Digital Lock locking position may be set in user units by setting up a units per rev conversion ratio with the command*SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[65]
Units		Encoder Counts or User units
Range		0 to 2^{31}-1
Default		6553

APCEnableRigidLock		
`Status% = APCEnableRigidLock()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command enables rigid digital lock, [60] = 1. In rigid digital lock mode, at the relative point the digital lock reference is selected, the profile generator will achieve the ratio of reference position and speed, within the constraints of the profile parameters, defined by commands *APCSetProfileAccelRate(Accel%)* [110], *APCSetProfileMaxSpeedClamp(MaxSpeed%)* [113]. When the feedback position and speed matches that of the ratio of the reference (within locking window parameters), the system is considered locked. The locking window parameters are defined by commands *APCSetDigLockLockingSpeed(LockSpeed%)* [64], and *APCSetDigLockLockingPosition(LockPosition%)* [65]. The profile generator deceleration rate in rigid lock mode is defined by the acceleration rate, *APCSetProfileAccelRate(Accel%)* [110]. Depending on the digital lock mode selected, defined by command *APCSetDigLockMode(Mode%)*[61], the profile generator will act accordingly when feedback position and speed matches that of the ratio of the reference.
APCReadPar		[60]
Units		NA
Range		Active or Inactive
Default		Inactive

APCDisableRigidLock		
`Status% = APCDisableRigidLock()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command disables rigid digital lock, and enables non-rigid digital lock, [60] = 0. In non rigid digital lock mode, the profile generator will achieve the ratio of reference speed, within the constraints of the profile parameters, defined by commands *APCSetProfileAccelRate(Accel%)* [110], *APCSetProfileDecelRate(Decel%)* [111], *APCSetProfileMaxSpeedClamp(MaxSpeed%)* [113]. When the feedback speed matches that of the ratio of the reference (within locking window parameter), the system is considered locked. The locking window parameter is defined by command *APCSetDigLockLockingSpeed(LockSpeed%)* [64]. Depending on the digital lock mode selected, defined by command *APCSetDigLockMode(Mode%)*[61], the profile generator will act accordingly when feedback position and speed matches that of the ratio of the reference.
APCReadPar		[60]
Units		NA
Range		Active or Inactive
Default		Active

5.2.5 Profile Generators

Figure 5-5



5.2.5.1 Main Profile Generator

APCEnableProfile		
`Status% = APCEnableProfile()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command enables the interlock for the internal Profile Generator [114] = 0. When enabled the Profile Generator bypass switch is controlled by the selected position reference function *APCSelectReference()*. Position (1), Speed (2) & Stop (0) Reference - The Profile Generator is enabled by this command. CAM (3) - This command has no effect as the Profile generator is always disabled. Digital lock (4) - The Profile generator enable is dependant on this command and the digital lock mode. The command *APCDisableProfile()* will disable/bypass the profile generator.
APCReadPar		[114]
Units		NA
Range		Active or inactive
Default		Active

APCDisableProfile		
`Status% = APCDisableProfile()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command forces the Profile Generator to be manually bypassed [114] = 1. Care should be taken when using this command as any fast/stepped changes from the selected source reference will be instantly seen by the position loop, as there are no ramps applied. This could cause damage to the applications mechanics.
APCReadPar		[114]
Units		NA
Range		Active or Inactive
Default		Inactive

APCSetProfileAccelRate		
`Status% = APCSetProfileAccelRate(AccelRate%)`		
Input Arguments	*AccelRate%*:	Signed 32-bit
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the main Profile Generator acceleration rate, [110]. When Digital Lock is selected, [100] = 4, with Rigid Lock mode, [60] = 1, this command will also set the deceleration rate of the Profile Generator. In SM-Apps Firmware =>V01.04.04 the main profile generator acceleration rate may be set in user units/s/s, by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[110]
Units		2^{32} counts / rev encoder over 250µs / 250µs or User units/s/s
Range		0 to 4473924
Default		22369 (equal to 1.25rpm / 250µs)

APCSetProfileDecelRate		
`Status% = APCSetProfileDecelRate(DecelRate%)`		
Input Arguments	*DecelRate%*:	Signed 32-bit
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the main Profile Generator deceleration rate [111]. In SM-Apps Firmware =>V01.04.04 the main profile generator deceleration rate may be set in user units/s/s, by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[111]
Units		2^{32} counts / rev encoder over 250µs / 250µs or User units/s/s
Range		0 to 4473924
Default		22369 (equal to 1.25rpm / 250µs)

APCSetProfileMaxSpeedClamp		
`Status% = APCSetProfileMaxSpeedClamp(MaxSpeed%)`		
Input Arguments	*MaxSpeed%*:	1 to 715,827,883
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the main Profile Generator maximum speed clamp [113]. In SM-Apps Firmware =>V01.04.04 the main profile generator profile maximum speed clamp may be set in user units/s, by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[113]
Units		2^{32} counts / rev encoder over 250µs or User units/s
Range		0 to 715827883
Default		48318382 (equal to 2700rpm)

5.2.5.2 Offset Profile Generator

APCSetSpeedOffset		
`Status% = APCSetSpeedOffset(Speed%)`		
Input Arguments	*Speed%*:	Signed 32-bit position
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the offset profile generator speed offset [95] and is summed with the main profile generator output. This is used to provide a speed trim/draw to the feedback axis with respect to the reference axis. When the Stop reference is selected, [100] = 0, the offset selector [102] will be set to 0 internally, and therefore the offsets will have no effect, and if profiled stop is selected [90] = 1, the offset profile acceleration and deceleration rates are ignored, and the main profile rates are used. In SM-Apps Firmware =>V01.04.04 the offset speed setpoint may be set in user units/s, by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[95]
Units		2^{32} counts / rev encoder over 250µs or User units/s
Range		-715827883 to 715827883
Default		0

APCSetPositionOffset		
`Status% = APCSetPositionOffset(Position%)`		
Input Arguments	*Position%*:	Signed 32bit position
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets offset profile generator position offset [93] and is summed with the main profile generator output. This can be used to phase advance/retard the feedback axis with respect to the reference axis. The value of `Position%` is entered in counts, and should be calculated with respect to the same resolution set for the feedback encoder, defined by the command *APCSetNumOfTurnsBits(TurnBits%)* [14]. When the Stop reference is selected, [100] = 0, the offset selector [102] will be set to 0 internally, and therefore the offsets will have no effect, and if profiled stop is selected [90] = 1, the offset profile acceleration and deceleration rates are ignored, and the main profile rates are used. In SM-Apps Firmware =>V01.04.04 the offset position setpoint may be set in user units with remainder, by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[93]
Units		Encoder Counts or User units
Range		-2^{31} to 2^{31} - 1
Default		0

APCSetOffsetRem		
`Status% = APCSetOffsetRem(Remainder%)`		
Input Arguments	*Remainder%*:	Signed 32bit position remainder
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the offset position setpoint remainder [175], and will be applied to the profile generator input when selected, using the command *APCSelectReference(1)* [102] = 1. This command is only available in SM-Apps Firmware =>V01.04.04, where the offset position setpoint may be set in user units with remainder by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[92]
Units		Encoder Counts or User Units
Range		-2^{31} to 2^{31} - 1
Default		0

APCSelectOffset		
`Status% = APCSelectOffset(OffSelect%)`		
Input Arguments	*OffSelect%*:	
	0:	Offsets disabled
	1:	Position offset [93] enabled
	2:	Speed offset [95] enabled
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
Description		This command sets which offset type is to be applied to the selected reference [102]. When the Stop reference is selected, [100] = 0, the offset selector will be set to 0 internally, and therefore the offsets will have no effect, and if profiled stop is selected [90] = 1, the offset profile acceleration and deceleration rates are ignored, and the main profile rates are used.
APCReadPar		[102]
Units		NA
Range		0 to 2
Default		0

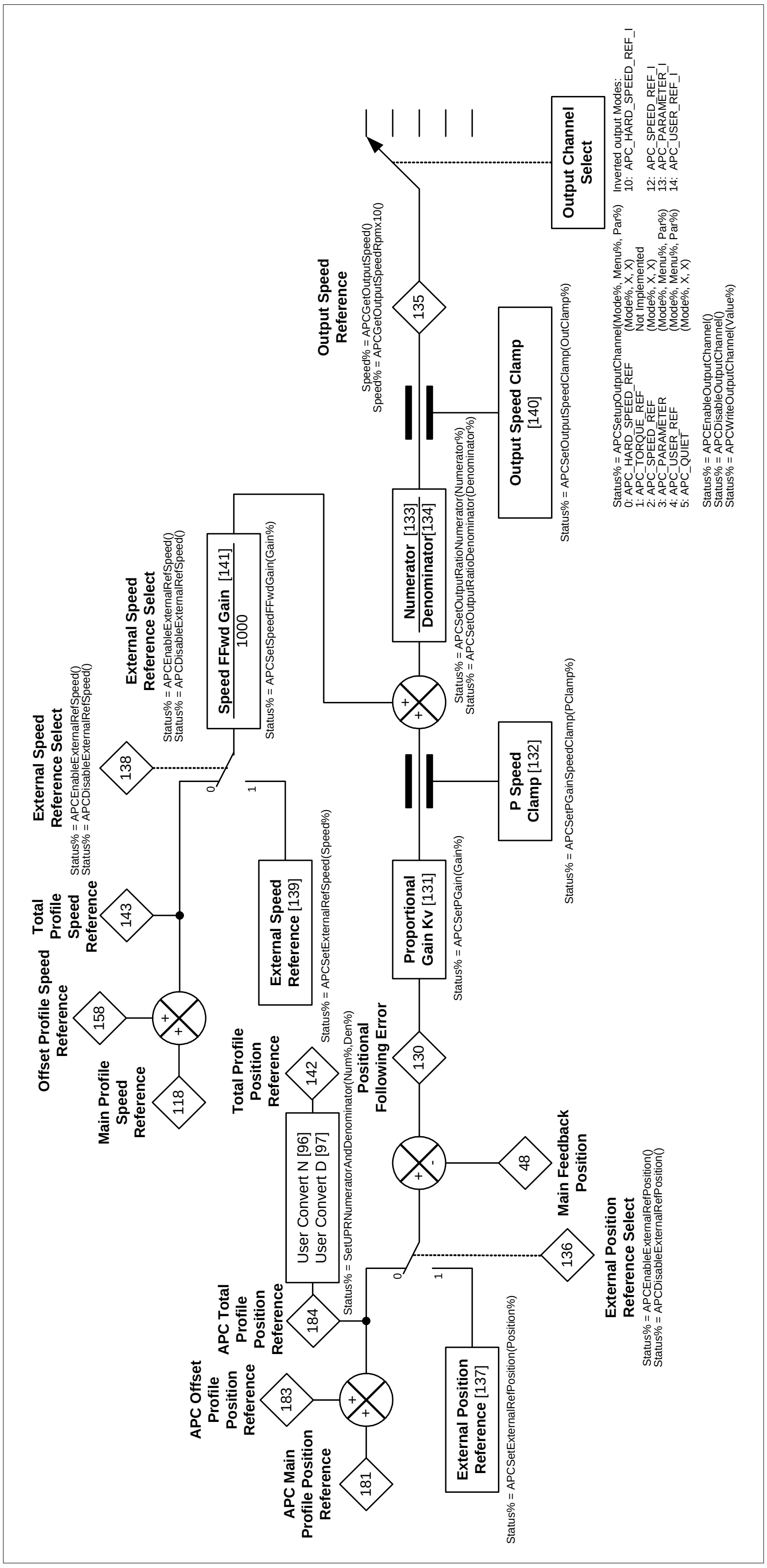
APCSetOffProfileAccelRate		
`Status% = APCSetOffProfileAccelRate(AccelRate%)`		
Input Arguments	*AccelRate%*:	Signed 32-bit
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the offset Profile Generator acceleration rate [150]. In SM-Apps Firmware =>V01.04.04 the offset profile generator acceleration rate may be set in user units/s/s, by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[150]
Units		2^{32} counts / rev encoder over 250µs / 250µs or User units/s/s
Range		0 to 4473924
Default		22369 (equal to 0.2s / 1000rpm)

APCSetOffProfileDecelRate		
`Status% = APCSetOffProfileDecelRate(DecelRate%)`		
Input Arguments	*DecelRate%*:	Signed 32-bit
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the offset Profile Generator deceleration rate [151]. In SM-Apps Firmware =>V01.04.04 the offset profile generator deceleration rate may be set in user units/s/s, by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. section 4.12 for more details.
APCReadPar		[151]
Units		2^{32} counts / rev encoder over 250µs / 250µs or User units/s/s
Range		0 to 4473924
Default		22369 (equal to 0.2s / 1000rpm))

APCSetOffProfileMaxSpeedClamp		
`Status% = APCSetOffProfileMaxSpeedClamp(MaxSpeed%)`		
Input Arguments	*MaxSpeed%*:	1 to 715,827,883
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the offset Profile Generator maximum speed clamp [153]. In SM-Apps Firmware =>V01.04.04 the offset profile generator profile maximum speed clamp may be set in user units/s, by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[153]
Units		2^{32} counts / rev encoder over 250µs or User units/s
Range		0 to 715827883
Default		48318382 (equal to 2700rpm)

5.2.6 Position Loop

Figure 5-6



APCSetPGain		
`Status% = APCSetPGain(Gain%)`		
Input Arguments	*Gain%*:	0 to 65535
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the proportional gain of the Position Loop controller in 0.01 radians / s / radian unit. If the position controller controls a motor that provides linear movement, the units of the proportional gain will be equivalent to meters / s / meter, or millimeters / s / millimeter. The gain is applied to the positional following error [130], and then converted in to a positional correction speed.
APCReadPar		[131]
Units		0.01 radians / s / radian unit
Range		0 to 65535
Default		2500

APCSetPGainSpeedClamp		
`Status% = APCSetPGainSpeedClamp(PClamp%)`		
Input Arguments	*PClamp%*:	0 to 715,827,883
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the Proportional Gain maximum speed clamp. This should normally be set to 10% of the maximum speed range [140] to allow for position correction. In SM-Apps Firmware =>V01.04.04 the proportional gain maximum speed clamp may be set in user units/s, by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[132]
Units		2^{32} counts / rev encoder over 250µs or User units/s
Range		0 to 715827883
Default		5368709 (equal to 300rpm)

APCEnableExternalRefSpeed		
`Status% = APCEnableExternalRefSpeed()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1	Operation successful
Description		This command enables the Position Loop feed forward reference to be changed, [138] = 1, from the Profile speed reference [116], to the External speed reference [139], set by the command *APCSetExternalRefSpeed(Speed%).*
APCReadPar		[138]
Units		NA
Range		Active or Inactive
Default		Inactive

APCDisableExternalRefSpeed		
`Status% = APCDisableExternalRefSpeed()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1	Operation successful
Description		This command disables the External speed reference [139], and enables Profile Position reference [116], [138] = 0, to be used as the Position Loop feed forward reference.
APCReadPar		[138]
Units		NA
Range		Active or Inactive
Default		Active

APCSetExternalRefSpeed		
`Status% = APCSetExternalRefSpeed(Speed%)`		
Input Arguments	*Speed%*:	Signed 32-bit
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the value of the external reference speed [139]. In SM-Apps Firmware =>V01.04.04 the external reference speed may be set in user units/s, by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[139]
Units		2^{32} counts / rev encoder over 250µs or User units/s
Range		-715827883 to 715827883
Default		0

APCEnableExternalRefPosition		
`Status% = APCEnableExternalRefPosition()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1	Operation successful
Description		This command enables position loop reference position to be changed, [136] = 1, from the Profile Position reference [117], to the External position reference [137], set by the command *APCSetExternalRefPosition(Position%).*
APCReadPar		[136]
Units		NA
Range		Active or Inactive
Default		Inactive

APCDisableExternalRefPosition		
`Status% = APCDisableExternalRefPosition()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed
	1	Operation successful
Description		This command disables the External position reference [137], and enables Profile Position reference [117], [136] = 0.
APCReadPar		[136]
Units		NA
Range		Active or Inactive
Default		Active

APCSetExternalRefPosition		
`Status% = APCSetExternalRefPosition(Position%)`		
Input Arguments	*Position%*:	Signed 32-bit
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the value of the external reference position [137], in counts. The value of `Position%` is entered in counts, and should be calculated with respect to the same resolution set for the feedback encoder, defined by the command *APCSetNumOfTurnsBits(TurnBits%)* [14]. In SM-Apps Firmware =>V01.04.04 the external reference position may be set in user units, by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[137]
Units		Encoder Counts or User units
Range		-2^{31} to $2^{31} - 1$
Default		0

APCSetSpeedFFwdGain		
`Status% = APCSetSpeedFFwdGain(Gain%)`		
Input Arguments	*Gain%*:	0 to 2000
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets position loop speed feed forward gain [141]. The speed feed forward reference will be multiplied by Gain% and divided by 1000 and can be used to provide high inertia positional dampening.
APCReadPar		[141]
Units		Number
Range		0 to 2000
Default		1000

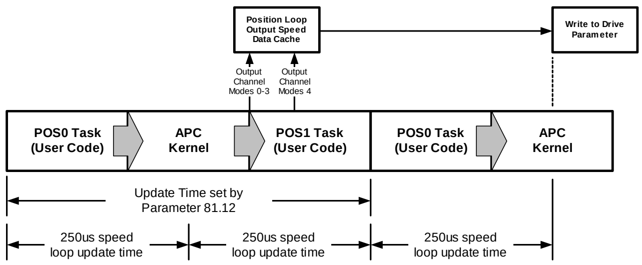
APCSetOutputRatioNumerator		
`Status% = APCSetOutputRatioNumerator(Numerator%)`		
Input Arguments	*Numerator%*:	Unsigned 31bit value to set as numerator
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets Position Loop numerator value [133]. The position loop output will be multiplied by the numerator value, before the position loop output speed clamp. It is used in conjunction with the position loop denominator *APCSetOutputRatioDenominator(Denominator%)*, and can be used where the feedback encoder is not directly attached to the drive motor i.e. These values can be set to compensate for any gearing between the slave motor shaft, and the feedback encoder.
APCReadPar		[133]
Units		Number
Range		0 to 2147483647
Default		1000

APCSetOutputRatioDenominator		
`Status% = APCSetOutputRatioDenominator(Denominator%)`		
Input Arguments	*Denominator%*:	Unsigned 31bit value to set as denominator
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets Position Loop denominator value [133]. The position loop output will be multiplied by the denominator value, before the position loop output speed clamp. It is used in conjunction with the position loop numerator *APCSetSpeedFFwdGain(Numerator%),* and can be used where the feedback encoder is not directly attached to the drive motor i.e. These values can be set to compensate for any gearing between the slave motor shaft, and the feedback encoder.
APCReadPar		[134]
Units		Number
Range		1 to 2147483647
Default		1000

APCSetOutputSpeedClamp		
`Status% = APCSetOutputSpeedClamp(Clamp%)`		
Input Arguments	*Clamp%*:	Signed 32-bit
Output Arguments	*Status%*:	
	0:	Operation failed
	1:	Operation successful.
Description		This command sets the value of the Output Speed Clamp [140]. This should equal the sum of the P Gain Speed Clamp [132], main profile generator output speed clamp [113], and the offset profile generator output speed clamp [153] (if used), set by the commands *APCSetPGainSpeedClamp(Clamp%)*, *APCSetProfileMaxSpeedClamp(Clamp%)*, and *APCSetOffProfileMaxSpeedClamp(Clamp%)*. In SM-Apps Firmware =>V01.04.04 the output speed clamp may be set in user units/s, by setting up a units per rev conversion ratio with the command *SetUPRNumeratorAndDenominator(Num%,Den%)*. See section 4.12 for more details.
APCReadPar		[140]
Units		2^{32} counts / rev encoder over 250µs or User units/s
Range		0 to 715827883
Default		53687091

APCSetupOutputChannel		
`Status% = APCSetupOutputChannel(Mode%, Menu%, Parameter%)`		
Input Arguments	*Mode%:*	
	0:	Write *Position loop Output Speed* to the Hard Speed Reference shortcut and *OutputSpeedRpmx10* to parameter #3.22.
	1:	Not allocated
	2:	Write *Position loop Output Speed* to the Speed Reference shortcut and *OutputSpeedRpmx10* to parameter #1.21.
	3:	Write *Position loop OutputSpeedRpmx10* to *Menu%.Parameter%*
	4:	Data provided by user at the end of the motion engine period is written to *Menu%.Parameter%* at the start of the next motion engine period.
	5:	Quiet mode. The APC will not automatically output data to the drive.
	Menu%:	Menu number to use if *Mode%*=3 or 4
	Parameter%:	Parameter number to use if *Mode%*=3 or 4
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
	-3:	Another process is modifying the object's data.
Description	*Mode%:*	
	0:	In this mode, the Position Loop Output will be automatically written to the Drives hard speed reference, Parameter #3.22. For the hard speed reference to control the drives speed, the Hard Speed reference has to be selected, by setting parameter #3.23 to 1. In this output mode `Menu%` and `Parameter%` do not have to be set.
	1:	`Mode%` should not be set to 1, as this feature is not available in this release.
	2:	In this mode, the Position Loop Output will be automatically written to the Drive preset speed reference 1, Parameter #1.21. For the preset speed reference to control the drives speed, the preset speed reference has to be selected, by setting parameter #1.14 to 3. In this output mode `Menu%` and `Parameter%` do not have to be set.
	3:	In this mode, the Position Loop Output will be automatically written the parameter of your choice, provided that it is not a protected or read only parameter. To select the drive parameter the output of the Position loop is to write to, `Menu%` and `Parameter%` must be set to the menu number, and parameter number respectively.
	4:	In this mode, the Output channel will write a user defined value to the parameter of your choice, provided that it is not a protected or read only parameter. To select the drive parameter the output channel is to write to, `Menu%` and `Parameter%` must be set to the menu number, and parameter number respectively. The value the output channel will write to is defined by the command *APCWriteOutputChannel(Value%)*, where *Value%* is a signed 32bit value
	5:	In this mode, the APC will not output any data to the drive.
APCReadPar		NA
Units		NA
Range		0 to 5
Default		5

APCEnableOutputChannel		
`Status% = APCEnableOutputChannel()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
	-3:	Another process is modifying the object's data.
Description		This command enables the APC's output channel to write data to the drive synchronously with the speed loop. The data will be sent 250µs with the next POS task update. If this Command is not active APC can not transfer output data to drive parameter selected by *APCSetupOutputChannel(Mode%, Menu%, Parameter%)*.



Note		**The output channel has to be configured before it is enabled. The output channel is configured by the command** *APCSetupOutputChannel***(Mode%, Menu%, Parameter%).**
APCReadPar		NA
Units		NA
Range		Active or Inactive
Default		Inactive

APCDisableOutputChannel		
`Status% = APCDisableOutputChannel()`		
Input Arguments	None	
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
	-3:	Another process is modifying the object's data.
Description		This command disables the APC's output channel, and prevents it from writing data to the drive.
APCReadPar		NA
Units		NA
Range		Active or Inactive
Default		Active

APCWriteOutputChannel		
`Status% = APCWriteOutputChannel(Value%)`		
Input Arguments	Value%	32bit signed value
Output Arguments	*Status%*:	
	0:	Operation failed.
	1:	Operation successful.
	-3:	Another process is modifying the object's data.
Description		This command sets the value sent to the drive, when the output channel is set to user reference, mode 4, defined by the command *APCSetupOutputChannel(4,Menu%,Parameter%)*.
APCReadPar		NA
Units		NA
Range		-2^{31} to 2^{31} - 1
Default		0

5.3 Conversion Functions

5.3.1 Embedded APC Converter

To assist the user to perform Position Control operations in familiar units, conversion from user units to APC kernel internal units, and from APC kernel internal to user units is provided. The user has to specify the number of units required for a single revolution of the motor (UPR), after which the position or distance, velocity, and acceleration can be specified in user units, user units/s, and user units/s/s respectively.

SetUPRNumeratorAndDenominator		
`Status% = SetUPRNumeratorAndDenominator(Numerator%,Denominator%)`		
Input Arguments	*Numerator%*	0 to 2^{31}-1
	Denominator%	1 to 2^{31}-1
Output Arguments	*Status%:*	
	0:	Operation failed
	1:	Operation successful
Description		This command sets up the APC user units to revolutions ratio used by the APC to run in user position, speed and acceleration etc. units. Numerator% must be set to the number of user units per number of revolutions set by Denominator%. These values are used to set the APC's internal conversion numerator and denominator [96] and [97], where: [96] = $2^{(32-[14])}$ * Denominator% [97] = Numerator% After setting Numerator% and Denominator%, all positions etc. read from the APC will be in user units, all speeds will be in user units/s, and all accelerations will be in user units/s/s. With the exception of the CAM reference, all other positioning references will be run in user units. See section 4.12 for further details.

GetUPRNumeratorAndDenominator		
`(Numerator%,Denominator%,Status%) = GetUPRNumeratorAndDenominator()`		
Output Arguments	*Status%*:	
	0:	The value APCPos% is too large to fit within 32-bit signed bounds. The value returned will be set to zero.
	1:	Operation successful.
	Numerator%	Returns the current value of the user unit conversion numerator.
	Denominator%	Returns the current value of the user unit conversion denominator.
Description		This command returns the current user unit conversion ratio numerator and denominator. See section 4.12 for more details.

UserToAPCPosition		
`(APCPos%,Status%) = UserToAPCPosition(UserPos%)`		
Input Arguments	*UserPos%*:	Signed 32-bit
Output Arguments	*Status%*:	
	0:	The value APCPos% is too large to fit within 32-bit signed bounds. The value returned will be set to zero.
	1:	Operation successful.
	APCPos%	The APC equivalent of the user position, if Status% is 1, otherwise zero.
Description		This command converts a user position in UserPos%, and will return an APC position in APCPos%.

APCToUserPosition		
`(UserPos%,Status%) = APCToUserPosition(APCPos%)`		
Input Arguments	*APCPos%*:	Signed 32-bit
Output Arguments	*Status%*:	
	0:	The value APCPos% is too large to fit within 32-bit signed bounds. The value returned will be set to zero.
	1:	Operation successful.
	UserPos%	The user equivalent of the APC position, if Status% is 1, otherwise zero.
Description		This command converts an APC position in APCPos%, and will return a user position in UserPos%.

UserToAPCVelocity		
`(APCVel%,Status%) = UserToAPCVelocity(UserVel%)`		
Input Arguments	*UserVel%*:	Signed 32-bit
Output Arguments	*Status%*:	
	0:	The value APCVel% is too large to fit within 32-bit signed bounds. The value returned will be set to zero.
	1:	Operation successful.
	APCVel%	The APC equivalent of the user velocity, if Status% is 1, otherwise zero.
Description		This command converts a user velocity (units/s) in UserVel%, and will return an APC velocity in APCVel%.

APCToUserVelocity		
`(UserVel%,Status%) = APCToUserVelocity(APCVel%)`		
Input Arguments	*APCVel%*:	Signed 32-bit
Output Arguments	*Status%*:	
	0:	The value APCVel% is too large to fit within 32-bit signed bounds. The value returned will be set to zero.
	1:	Operation successful.
	UserVel%	The User equivalent of the APC velocity, if Status% is 1, otherwise zero.
Description		This command converts an APC velocity in APCVel%, and will return a User velocity (units/s) in UserVel%.

UserToAPCAcceleration		
`(APCAccel%,Status%) = UserToAPCAcceleration(UserAccel%)`		
Input Arguments	*UserAccel%*:	Signed 32-bit
Output Arguments	*Status%*:	
	0:	The value APCVel% is too large to fit within 32-bit signed bounds. The value returned will be set to zero.
	1:	Operation successful.
	APCAccel%	The APC equivalent of the user acceleration, if Status% is 1, otherwise zero.
Description		This command converts a user acceleration (units/s/s) in UserAccel%, and will return an APC acceleration in APCAccel%.

APCToUserAcceleration		
`(UserAccel%,Status%) = APCToUserVelocity(APCAccel%)`		
Input Arguments	*APCAccel%*:	Signed 32-bit
Output Arguments	*Status%*:	
	0:	The value UserAccel% is too large to fit within 32-bit signed bounds. The value returned will be set to zero.
	1:	Operation successful.
	UserAccel%	The User equivalent of the APC acceleration, if Status% is 1, otherwise zero.
Description		This command converts an APC acceleration in APCAccel%, and will return a User acceleration (units/s/s) in UserAccel%.

5.3.1.1 Notes on Rounding

In most cases it will be possible to convert a user value (e.g. position) to an internal APC position and back again and retain exactly the same value. For example:

UPR N = 5000

UPR D = 1

Number Of Turns Bits = 9

User position of 1000 gives APC position of 1677722 via the "UserToAPCPosition" function block.

(This is actually 1677721.6 but gets rounded to 1677722)

APC position of 1677722 gives user position of 1000 via the "APCToUserPosition" function block.

(This is actually 1000.0002384… but gets rounded to 1000)

As the APC kernel uses integer maths, in each conversion the fractional part is rounded to the nearest whole unit. This can sometimes lead to unexpected errors.

If we perform the calculation above, with number of turns bits set to 9, feeding in User Position of 1000 each time, but with differing values for UPR we obtain the following:

Table 5-1

UPR	Ideal APC Position	Rounded APC Position	Ideal user position (converted back from rounded APC Position)	Rounded user position (converted back from rounded APC Position)
5,000	16777221.6	1677722	1000.000	1000
50,000	1677722.16	167772	999.999	1000
500,000	167772.216	16777	999.987	1000
5,000,000	16777.2216	1678	1000.165	1000
50,000,000	1677.72216	168	1001.358	1001
500,000,000	167.772216	17	1013.278	1013
5,000,000,000	16.7772216	2	1192.092	1192

Note that in the final 3 example the error is greater than a half, which results in an error in the result, even after rounding. We do not get the same value back.

NOTE **The notes on rounding in this section are true for all systems using rounding, however with the new user unit conversion system implemented in SM-Application firmware =>V01.04.04, user unit remainder has been added to prevent any loss of positional information when converting to user units. See section 4.12** *APC Operation in User Units* **.**

This problem occurs when the units per revolution is far greater than the position being converted, and can be improved by decreasing the number of turns bits to provide better resolution:

Number Of Turns Bits = 5

Table 5-2

UPR	Ideal APC Position	Rounded APC Position	Ideal user position (converted back from rounded APC Position)	Rounded user position (converted back from rounded APC Position)
5,000	16777221.6	26843545.6	1000.000	1000
50,000	1677722.16	2684354.56	1000.000	1000
500,000	167772.216	268435.456	999.998	1000
5,000,000	16777.2216	26843.5456	1001.016	1000
50,000,000	1677.72216	2684.35456	999.867	1000
500,000,000	167.772216	268.435456	998.377	998
5,000,000,000	16.7772216	26.8435456	1005.828	1006

The improved resolution in the second example reduces the effect of the rounding, so that we can have a greater UPR / Input Value ratio before rounding errors occur.

5.3.2 User Defined Unit Converter

Note that this is not intended exclusively for use with the APC, but can be used for any purpose. For example millimeters to inches conversion and back again can be performed by setting the numerator to 254 and the denominator to 10. Converting forwards will scale by 25.4, which will convert inches to millimeters. Converting backwards will convert millimeters to inches.

SetConverterNumerator		
`SetConverterNumerator(Numerator%)`		
Input Arguments	*Numerator%*:	Signed 32-bit
Description		This command sets the numerator value for use as a scaling factor by the User Defined Unit Converter.

SetConverterDenominator		
`SetConverterDenominator(Denominator%)`		
Input Arguments	*Denominator%*:	Signed 32-bit
Description		This command sets the denominator value for use as a scaling factor by the User Defined Unit Converter.

ConvertForwards		
`Output%=ConvertForwards(Input%)`		
Input Arguments	*Input%*:	Signed 32-bit
Output Arguments	*Output%:*	Signed 32-bit
Description		This command takes a 32-bit integer value from the input variable, `Input%`, and multiplies it by the numerator/ denominator. The inverse of this function is APCConvertBackwards.

ConvertBackwards		
`Output%=ConvertBackwards(Input%)`		
Input Arguments	*Input%*:	Signed 32-bit
Output Arguments	*Output%:*	Signed 32-bit
Description		This command takes a 32-bit integer value from the input variable, `Input%`, and multiplies it by the denominator/ numerator. The inverse of this function is APCConvertForwards.

5.3.3 Word Manipulation Function Blocks

UnsignedTopWord		
`Output%=UnsignedTopWord(Input%)`		
Input Arguments	*Input%*:	Signed 32-bit
Output Arguments	*Output%:*	Un-signed 32-bit
Description		This command takes a 32-bit integer value from the input variable, `Input%`, and converts the upper 16-bits to an unsigned 16-bit value, in the lower 16-bits of the output variable `Output%`. For the example `Output%=UnsignedTopWord(0x01234567)` the result in `Output%` equals (0x00000123), a positive number. For the example `Output%=UnsignedTopWord(0x98768765)` the result in `Output%` equals (0x0009876), a positive number.

SignedTopWord		
`Output%=SignedTopWord(Input%)`		
Input Arguments	*Input%*:	Signed 32-bit
Output Arguments	*Output%:*	Un-signed 32-bit
Description		This command takes a 32-bit integer value from the input variable, `Input%`, and converts the upper 16-bits to a Signed 16-bit value, in the lower 16-bits of the output variable `Output%`. For the example `Output%=UnsignedTopWord(0x01234567)` the result in `Output%` equals (0x00000123), a positive number. For the example `Output%=UnsignedTopWord(0x98768765)` the result in `Output%` equals (0xFFFF9876), a negative number.

UnsignedBottomWord		
`Output%=UnsignedBottomWord(Input%)`		
Input Arguments	*Input%*:	Signed 32-bit
Output Arguments	*Output%:*	Un-signed 32-bit
Description		This command takes a 32-bit integer value from the input variable, `Input%`, and converts the lower 16-bits to an Unsigned 16-bit value, in the lower 16-bits of the output variable `Output%`. For the example `Output%=UnsignedBottomWord(0x01234567)` the result in `Output%` equals (0x00004567), a positive number. For the example `Output%=UnsignedBottomWord(0x98768765)` the result in `Output%` equals (0x00008765), a positive number.

SignedBottomWord		
`Output%=SignedBottomWord(Input%)`		
Input Arguments	*Input%*:	Signed 32-bit
Output Arguments	*Output%:*	Un-signed 32-bit
Description		This command takes a 32-bit integer value from the input variable, `Input%`, and converts the lower 16-bits to a signed 16-bit value, in the lower 16-bits of the output variable `Output%`. For the example `Output%=SignedBottomWord(0x01234567)` the result in `Output%` equals (0x00004567), a positive number. For the example `Output%=SignedBottomWord(0x98768765)` the result in `Output%` equals (0xFFFF8765), a negative number. This Function can be used to sign extend a word.

PutTopWord		
`Output%=PutTopWord(Input1%,Input2%)`		
Input Arguments	*Input%*:	Signed 32-bit
Output Arguments	*Output%:*	Un-signed 32-bit
Description		This command takes the lower 16 bits of the 32-bit integer value from the input variable, `Input1%`, and puts them in to the upper 16bits of the output variable, `Output%`. It also takes the lower 16 bits of the 32-bit integer value from the second input variable, `Input2%`, and puts them in to the lower 16bits of the output variable, `Output%.` For the example `Output%=PutTopWord(0x00000068,0x00000057)` the result in `Output%` equals (0x00680057), a positive number. For the example `Output%=PutTopWord(0xFFFFFFF7,0x00000057)` the result in `Output%` equals (0xFFF70057), a negative number.

PutBottomWord		
`Output%=PutBottomWord(Input1%,Input2%)`		
Input Arguments	*Input%*:	Signed 32-bit
Output Arguments	*Output%:*	Un-signed 32-bit
Description		This command takes the lower 16 bits of the 32-bit integer value from the input variable, `Input1%`, and puts them in to the lower 16bits of the output variable, `Output%`. It also takes the upper 16 bits of the 32-bit integer value from the second input variable, `Input2%`, and puts them in to the upper 16bits of the output variable, `Output%.` For the example `Output%=PutBottomWord(0x00000068,0x00000057)` the result in `Output%` equals (0x00000068), a positive number. For the example `Output%=PutBottomWord(0xFFFFFFF7,0x57043291)` the result in `Output%` equals (0x5704FFF7), a positive number.

6 Getting Started

6.1 Hardware Selection

6.1.1 Motor and Drive

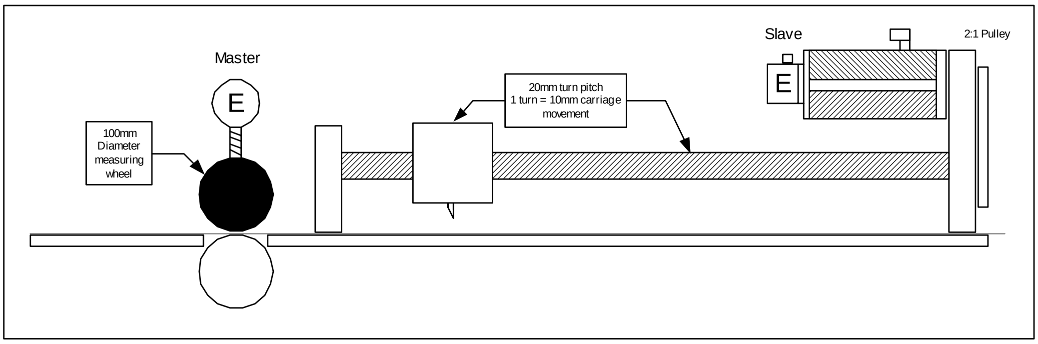
The drive and motor combination chosen for the application, must be selected such that the drive used has sufficient current capability, that it will not go into current limit under the most demanding dynamic moves, the application is likely to incur. Under current limit conditions, the drive will not be able to follow the demanded speed from the application motion profile.

To see if the drive is going into current limit, check if parameter 10.09 is changing to 1 or ON.

6.1.2 Feedback and Reference encoder

Before the application encoders can be selected the application accuracy/resolution of the system needs to be decided, then the encoder required can be calculated. A rotational and linear example is discussed below:

Figure 6-1



1. Rotational (Master)

   In Figure 6-1 there is a measuring wheel 100mm in diameter. The circumference of the wheel is:

   Circumference = π*Diameter = π*100 = 314.15

   If we use a resolution/accuracy of 0.1mm units there are:

   314.15 / 0.1 = 3141.5 Fine units per revolution

   A good rule of thumb is to use an encoder that can produce 20 counts per fine resolution unit **minimum** (the more counts the better):

   Encoder counts per revolution = 20 * 3141.5 = 62830

   The closest standard encoder value rounded up is 65536 counts per revolution, or a 16bit resolution encoder. This encoder could be a SinCos, SSI or a high resolution quadrature (16384ppr).

2. Linear (Slave)

   In Figure 6-1 there is a ball screw with a turn pitch of 20mm, connected to a 2:1 pulley, which is in turn connected to the motor. The turn pitch at the motor is 10mm, so Keeping the same fine resolution of 0.1mm:

   10/0.1 = 100 Fine units per revolution

   Using the same rule of thumb at 20 counts per fine resolution unit:

   20*100 = 2000

The closest standard value rounded up is 2048 counts per revolution, or a 11bit resolution encoder

From a practical point of view the slave encoder resolution would have to be raised to 14bit = 16384 counts per revolution = 4096 Pulses per revolution, as this is the minimum encoder size that should be used on a servo motor, and is the standard value quadrature encoder supplied by Control Techniques.

The Master encoder resolution should also be raised by the same number of bits to 19bit, as it is important to keep the ratio of the number of counts per unit, between the master and slave at similar value. This is because if the master encoder has less resolution than the slave, like a 10:1 difference, for every 1 count moved by the master the slave will have to move by 10 counts. This means that the system will act like an amplifier with a high gain, which can produce noisy or even unstable operation.

6.1.2.1 System Resolution

The resolution of the system is defined by the type of encoder used in the application, and the number of turns bits set in the APC by the command *APCSetNumOfTurnsBits*. In general, incremental encoders give the lowest resolution, and SinCos Encoders give the highest resolution available, however the final resolution available is down to the individual encoder chosen for the application, as encoder resolution can vary.

6.1.2.2 Absolute or Relative

There are two categories of encoder Relative or Absolute. Relative encoders only provide positional information within one revolution, whereas Absolute encoders provide positional information within a revolution, and turns information, and can keep track of the position even when there is no power.

Absolute “single” and “multiturn” encoders, have an internal mechanical encoder, which keeps track of the position even when the power is turned off. These encoders transmit their position data to the drive on power up, so that the current position, or position and number of turns is known straight away.

6.1.2.3 Encoder Pro’s and Con’s

Table 6-1 below shows a comparison of Resolvers, Incremental encoders, and SinCos encoders:

Table 6-1

Parameter	Resolver	Incremental Encoder	SinCos Encoder
Mechanical Shock and Vibration resistance	Excellent	Average	Average
Temperature rating	up to 150°C (Typical)	up to 105°C (Typical)	up to 105°C (Typical)
Resolution	12 to 16bit (Typical)	12 to 16bit (Typical)	up to 21bit (Typical)
Low speed operation	Average	Average	Excellent

6.1.2.4 APC Reference Encoder

A Reference Source encoder is only required for the CAM and Digital Lock APC references. The Reference source encoder position, may come from a standard encoder or from software sources via the User Program reference, like CTSync. For more information on CTSync refer to section 4.10 *CTSync* .

6.1.3 SM-Applications or SM-Applications Lite

Refer to section 3.7.1 *SM-Applications Module* and section 3.7.2 *SM-Applications Lite module* , for all available functionality and features.

6.2 Drive and Encoder setup

The drive must be setup and tuned correctly, so that the APC can give the best performance. In particular, It is essential that the drives Current loop and Speed loop gains are set correctly, as these directly affect the systems dynamic performance. Chapters 4, 7, and 8 of the Unidrive SP User guide (supplied on the CD with Unidrive SP), contains a comprehensive guide on how to connect the supply, motor and encoder, together with a commissioning and tuning guide.

Once the drive and encoder has been commissioned it is important to set up the following parameters, either on the drive or in program code in the initial task:

Table 6-2

Parameter	Description	Value
1.06	Drive maximum speed	Application dependant (set to 3000rpm from default)
1.07	Drive minimum speed	0
1.10	Bipolar reference select	1
1.14	Use preset speed reference	3
1.15	Preset speed reference 1 selected	1
2.02	Disable drive ramps	0
2.04	Use fast decel ramps	0
6.01	Stop with no ramps	2

NOTE Parameter 1.06 will override the APC’s output speed clamp which is also set to 3000rpm from default

In user code this would look like:

Figure 6-2

```
Initial{
//Drive setup Parameters
#01.06 = 3000
#01.07 = 0
#01.10 = 1
#01.14 = 3
#01.15 = 1
#02.02 = 0
#02.04 = 0
#06.01 = 2
}//Initial
```

NOTE The code above has been set up assuming the APC’s output channel is written to preset speed reference 1, alternatively the output channel could be written to the hard speed reference, in which case parameters 1.10, 1.14, 1.15 and 2.02 do not need to be set.

6.3 SM-Applications Setup for APC

The parameters in this section are referred to as #81.xx. They are aliases to menu 15, 16 or 17, depending on which slot the SM-Applications module is fitted to. When writing code for the SM-Applications module, using #81.xx ensures that the module can be fitted to any slot, and the correct corresponding slot menu will be updated/changed.

Table 6-3 below shows the parameters which must be setup to allow the APC to function as the user requires:.

Parameter	Description	Units	Range	Default
81.12	POS Task scheduling rate	Milliseconds	0: Disable 1: 0.25ms 2: 0.5ms 3: 1ms 4: 2ms 5: 4ms 6: 8ms	0
81.13	Auto-run enable (This must be set manually)	NA	0: Disable 1: Enable	0
81.14	Global trip enable	NA	0: Disable 1: Enable	0
81.16	Encoder data update rate (APC and Menu 90)	NA	0: Every 0.25ms 1: Every POS Task 2: Every Clock Task 3: Never	0
81.17	Parameter overrange trip enable	NA	0: Disable 1: Enable	0
81.38	Disable drive trip on APC Runtime Error (prevents Tr81 if required)	NA	0: Disable 1: Enable	0

Table 6-3

To get a basic APC program running, these parameters should be set as shown in Table 6-4 below, and can be adjusted as required later:

Parameter	value
81.12	3
81.13	1
81.14	1
81.16	1
81.17	0
81.38	1

Table 6-4

In user code this would look like Figure 6-3:

Figure 6-3

```
Initial{
//SM-Applications setup Parameters.
#81.12 = 3
#81.14 = 1
#81.16 = 1
#81.17 = 0
#81.38 = 1
REINIT
}//Initial
```

6.4 User Program

An APC application program generally consists of a:

- **Notes Task** - which explains broadly how the code works.
- **Initial Task** - which sets up the SM-Applications module, drive parameters, alias' and the APC.
- **Background Task** - Which is the lowest priority task, and handles operations which are not time critical e.g. Limits, scaling, indication etc.
- **POS0 and POS1Tasks** - Which run before and after the APC respectively. These tasks can be used to manipulate the APC and related position information, before and after the APC runs. Only time critical operations should be performed in this task.

6.4.1 Basic User Program.

The the program code in *section 7* shows working code examples, where every APC setup command has been called, however if the default APC setup is acceptable, then there are very few commands that actually have to be used to get a simple setup running.

From default, all of the key APC setup parameters have sensible values entered; these are briefly described below:

- The APC is in Relative mode - This means that after a reset, or the APC is disabled, or on power up, the Integrated position counters are reset to zero instead of the absolute position.
- Source reset on disable is active - This means that APC is allowed to perform a reset when the user calls one in the code.
- The reset position offset is set to 0 - This means that if the user calls a reset, no extra position will be added to the relative position.
- The main feedback and reference integrated counter positions are not inverted
- The number of turns bits is set to 16 - This means that the source encoder resolution that the APC sees, has been set to 16bit or 65536 counts per revolution.
- The Stop reference has been set to Profiled stop. - This means that when the Stop reference is selected to stop the axis, it will do it using the profile generator ramps.
- The Acceleration and Deceleration rate has been set to 22369 2^{32} counts per rev/250µs/250µs or 1.25rpm/250µs which is equal to 5000rpm/s.
- The Profile maximum speed is set to 48318382 2^{32} counts per rev/250µs, or 2700rpm
- The Profile generator is enabled from default - This means that if a reference needs the profile generator it will use the profile maximum speed clamp and the acceleration/deceleration rate set.
- The position loop proportional gain is set to 25 times - This means that any position error is multiplied by 25 and fed back into the position loop to ensure that the reference speed or position is maintained quickly.
- The position loop proportional gain speed clamp is set to 300rpm - This means that if the error would exceed 300rpm, it will be clamped to 300rpm. This improves stability with high proportional gains.

- All of the ratio's within the APC are set to 1:1 - This includes the Digital Lock, CAM and Position Loop numerator and denominators, which are both set to 1000.

Refer to the *Functional Description* or *APC Command Descriptions* sections for more details.

6.4.2 Enable APC

The APC Run mode should be interlocked with drive parameter 1.11, drive reference on, in the POS0 task. To ensure the APC will not be able to build up an excessive following error whilst the drive is tripped, disabled, or if the run signal was lost for any reason. This is particularly important when using references like CAM and Digital lock as an excessive following error would cause an instantaneous jump in position, when the drive is reset, or the run signal re-applied, which could damage the applications mechanics. Whilst in the APC is in the disabled state the, APC tracks the feedback and reference positions.

In user code this may look like:

Figure 6-4

```
POS0{
//Drive Reference on interlock
IF #01.11 = 1 THEN
    RunModeStatus% = APCSetRunMode(APC_ENABLE)//APC fully functional
ELSE
    RunModeStatus% = APCSetRunMode(APC_DISABLE)//Only APC source counters active
ENDIF
}//POS0
```

6.4.3 Output Channel

The APC has a destination configurable output speed channel which must be configured and enabled, so that the APC can control the drive. The Output channel is not configured and is disabled from default. Using the output channel ensures drive parameters are updated at the fastest rate possible

The example code used throughout this section has been written assuming that the output channel will write to preset speed 1 (Parameter 1.21). Figure 6-5 below shows a code example on how the output channel may be configured and enabled.

Figure 6-5

```
Initial{
//Configure then enable the APC Output Channel (writing to 1.21)
APCChStatus1% = APCSetupOutputChannel(2,0,0)
APCChStatus2% = APCEnableOutputChannel()
}//Initial
```

Refer to the *Functional Description* or *APC Command Descriptions* sections for more details.

6.4.4 Feedback and Reference Source Encoders

The Reference and Feedback source encoders must be defined for the APC to function correctly. The User can choose from the drive encoder input, an encoder option in one of the slots, a user program reference, or no encoder. It is recommended that the Feedback source encoder is directed to the drive encoder. The example code in this section has been written assuming that the Reference source encoder data will be provided by a virtual master counter, which has been incorporated into the CAM and Digital Lock reference examples.

Figure 6-6 below shows a code example on how the Reference and Feedback source encoders may be configured:

Figure 6-6

```
Initial{
//Configure the APC Feedback and Reference Source encoders
SourceStatus1% = APCSetFeedbackSource(APC_DRIVE_ENC) // Drive encoder selected
SourceStatus2% = APCSetReferenceSource(APC_USER_ENC) // User Program Selected
RunModeStatus% = APCSetRunMode(APC_DISABLE) // Primes the position counters on
startup or reset
}//Initial
```

Refer to the *Functional Description* or *APC Command Descriptions* sections for more details.

6.4.5 Stop Reference

The Stop reference allows the user to decelerate the axis, no matter what reference was previously selected, in a controlled manner.

The Stop reference has two modes of operation, which are Profile Stop and Instant Stop. Profile Stop is enabled from default.

Code showing the use of the Stop reference has been combined with the other references example code, like Speed and Position reference.

6.4.6 Position Reference

The Position reference allows the user to make simple profile controlled, point to point, indexing moves. The resolution is set to 65536 counts per turn from default, and can be easily converted to a custom number of units per turn, or units per count

Example code using the Position reference is shown in Figure 6-7 below:

Figure 6-7

```
Background{
// Position Setpoint Scaling from 0.1revs entered in parameter 18.11 to counts
PosSetpointCnts% = MULDIV(#18.11, 65536, 10)

//Send Position Reference to APC
PosRefStatus% = APCSetPositionSetPoint(PosSetpointCnts%)

// Current Position Indication from counts to 0.1revs
FeedbackPosCnts% = APCReadPar(APC_FB_POS)
#18.12= MULDIV(FeedbackPosCnts%, 10, 65536)

}//Background
POS0{
//18.31 selects either the Stop or Position Reference
IF #18.31 = 1 THEN
    RefSelectStatus% = APCSelectReference(APC_POSITION_REF)
ELSE
    RefSelectStatus% = APCSelectReference(APC_STOP_REF)
ENDIF
}//POS0
```

Refer to the *Functional Description* or *APC Command Descriptions* sections for more details.

6.4.7 Speed Reference

The Speed reference allows the user to perform jogging and homing type applications.

The Speed setpoint is set in 2^{32} counts / rev encoder over 250μs, however these units are easily converted into rpm or any other preferred unit.

A code example using the Speed reference is shown in Figure 6-7:

Figure 6-8

```
Background{
// Speed Setpoint Scaling from 0.1rpm entered in parameter 18.13 to Internal units
SpeedSetpointInternal% = MULDIV(1073741824,(#18.13/10), 60000)

//Send Speed Reference to APC
SpeedRefStatus% = APCSetSpeedSetPoint(SpeedSetpointInternal%)

}//Background
POS0{
//18.32 selects either the Stop or Speed Reference
IF #18.32 = 1 THEN
    RefSelectStatus% = APCSelectReference(APC_SPEED_REF)
ELSE
    RefSelectStatus% = APCSelectReference(APC_STOP_REF)
ENDIF
}//POS0
```

Refer to the *Functional Description* or *APC Command Descriptions* sections for more details.

6.4.8 CAM Reference

The CAM reference provides varied motion from a continuous reference motion. All of the position changes that the CAM reference produces, are in numbers of feedback encoder counts that must be moved, forwards or backwards, with respect to a number of reference encoder counts at a given speed. For a more detailed description of how the CAM reference functions see *section 4.7 CAM Reference .*

Example code using the CAM reference in single shot or continuous run mode, is shown in Table 6-7 below, including a virtual master reference counter:

Figure 6-9

```
Initial{

//Initialise Virtual master counter variable
MasterCounter% = 0

//Dimension CAM data arrays
DIM InArray%[3]
DIM OutArray%[3]

//Enter Position data into CAM Arrays
InArray%[0] = 65536 // 1 turn forwards
InArray%[1] = 65536 // 1 turn forwards
InArray%[2] = 65536 // 1 turn forwards

OutArray%[0] =  5536 // 1 turn forwards
OutArray%[1] =  131072 // 2 turns forwards
OutArray%[2] = -196608 // 3 turns backwards (back to start position)

//Initialise CAM arrays
CAMIniStatus% = APCCAMInitialise1(InArray%, OutArray%)

//Set Interpolation type
CAMIntStatus% = APCSetCAMInterpolationMode(APC_COSINE_CAM)

//Set CAM start point and size
CAMStartStatus% = APCSetCAMStartIndex(0) // start at CAM array element 0
CAMSizeStatus% = APCSetCAMSize(3) // 3 pairs of CAM array elements
}// Initial

Background{
// Current CAM Index/Segment Indication
#18.15 = APCReadPar(APC_CAM_INDEX)

}// Background
```

```
POS0{
//18.33 selects either the Stop or CAM Reference
IF #18.33 = 1 THEN
    RefSelectStatus% = APCSelectReference(APC_CAM_REF)
ELSE
    RefSelectStatus% = APCSelectReference(APC_STOP_REF)
ENDIF

//Set single shot or continuous CAM
IF #18.34 = 1 THEN
    CAMSSStatus% = APCEnableCAMSingleShot()
ELSE
    CAMSSStatus% = APCDisableCAMSingleShot()
ENDIF

// Virtual Master counter
// Counts per POS0 Task sample (1ms) = (RPM * Counts per Rev) / (60 * 1000)
CountsPerSample% = MULDIV(#18.16, 65536, 60000)//18.16 = Virtual master speed in rpm

// Virtual Counter
MasterCounter% = MasterCounter% + CountsPerSample%

// Split up Virtual Master Counter data ready for the APC User Program Reference
Turns% = UnsignedTopWord(MasterCounter%)
Position% = UnsignedBottomWord(MasterCounter%)
PositionFine% = 0

// Send Virtual Master counter data to the APC User Program Reference
UserRefStatus% = APCSetReferencePosition(Turns%, Position%, PositionFine%)

}// POS0
```

Refer to the *Functional Description* or *APC Command Descriptions* sections for more details.

NOTE The drive parameters used in POS0, have been used for demonstration purposes only. In real applications, access to drive parameters should be limited as much as possible. Where drive parameter access is required in real time tasks like POS0, variables should be associated in the background task, and then used in the real time code.

6.4.9 Digital Lock

The Digital Lock Reference provides a method of synchronising a Slave (feedback) axis to a Master (reference) axis. Example code using the Digital Lock reference in rigid or non-rigid lock, with variable output ratio is shown in Figure 6-10:

Figure 6-10

```
Initial{

//Initialise Virtual Master Counter variable
MasterCounter% = 0

//Initialise digital lock ratio parameters to set a 1:1 ratio
#18.17 = 1000
#18.18 = 1000
}// Initial

Background{
//Set rigid or non-rigid Digital Lock
IF #18.36 = 1 THEN
    RigidLckStatus% = APCEnableRigidLock() // Sync Speed and Position before locking
ELSE
    RigidLckStatus% = APCDisableRigidLock() // Sync Speed only before locking
ENDIF

// Set the Digital Lock Ratio
NumeratorStatus% = APCSetDigLockRatioNumerator(#18.17)
DenominatorStatus% = APCSetDigLockRatioDenominator(#18.18)

}// Background
POS0{
//18.35 selects either the Stop or Digital Lock Reference
IF #18.35 = 1 THEN
    RefSelectStatus% = APCSelectReference(APC_DIG_LOCK_REF)
ELSE
    RefSelectStatus% = APCSelectReference(APC_STOP_REF)
ENDIF

// Virtual Master counter
// Counts per POS0 Task sample (1ms) = (RPM * Counts per Rev) / (60 * 1000)
CountsPerSample% = MULDIV(#18.16, 65536, 60000)// 18.16 = Virtual master speed in rpm

// Virtual Counter
MasterCounter% = MasterCounter% + CountsPerSample%

// Split up Virtual Master Counter data ready for the APC User Program Reference
Turns% = UnsignedTopWord(MasterCounter%)
Position% = UnsignedBottomWord(MasterCounter%)
PositionFine% = 0

// Send Virtual Master counter data to the APC User Program Reference
UserRefStatus% = APCSetReferencePosition(Turns%, Position%, PositionFine%)

}// POS0
```

Refer to the *Functional Description* or *APC Command Descriptions* sections for more details.

NOTE

The drive parameters used in POS0, have been used for demonstration purposes only. In real applications, access to drive parameters should be limited as much as possible. Where drive parameter access is required in real time tasks like POS0, variables should be associated in the background task, and then used in the real time code.

6.5 Using the example code in this section

Below are some tips on how to use the example code in this section:

- The example code in this section has been laid out so that the user can construct a program by copy and pasting the code examples from the PDF manual, directly into the DPL editor in Sypt Workbench. The text tool in Adobe® Acrobat, can be used to select the text, then copy and paste as normal.
- Before pasting the code, insert a Initial task, POS0 task, and a Notes task if required, so that the code can go directly into the relevant task.
- Within each example, there may be code for more than 1 task e.g. Background and POS0, therefore code for the Background must be pasted into the Background task in the DPL editor.
- Where there are multiple code entries for a task, any new code for a task should be pasted after the previous, in the order the code is given in this section.

6.6 Final Performance Checks.

If the performance achieved is not as expected, the following performance check will determine where the tuning problem lies.

6.6.1 Checking Speed Loop

This should be done by monitoring analogue output 1 with an oscilloscope. The following parameters must be set in order to monitor the speed from analogue output 1:

- 7.19 = 3.02
- 7.21 = H.Spd / 3
- 10.38 = 100 or press reset (to activate changes to the analogue inputs)

Connect the oscilloscope to pins 9 and 11(0V).

NOTE If analogue output 1 does not provide a sufficient output voltage for the demand speed, lower drive parameter 1.06 till it is approximately 10% above the demand speed. This will give more volts per rpm, as the scaling for the analogue output is based on the value of parameter 1.06.

To monitor the drives speed loop performance the P gain for the position loop within the APC must be set to zero by using the command APCSetPGain(Gain%) in the user code, where Gain% = 0. Removing the P gain means that APC no longer regulates the position of the motor, so the drive is regulating it's self.

Ensure the speed loop can control the load, by setting the fastest acceleration rate the application requires, and applying a step change in demand velocity, whilst monitoring the performance using the oscilloscope. If it can not control the load sufficiently, follow the instructions in the Optimisation section in the Unidrive SP user guide.

6.6.2 Checking Position loop

Once the Speed loop is tuned, the Position loop P gain is re-applied and tuned to reduce the following error, shown by read parameter [130]. If a more dynamic reduction of following error under acceleration is required, either reduce the acceleration rate, or create a torque feed forward; see *Torque Feed forward* note in the *Functional Description* section.

If an over shoot which can not be tuned out by the P term is present, it is likely that the load used has a high inertia. The compensation setup shown in the *Compensation For Overshoot With High Inertia Load* application note resolves this problem.

7 Program Examples

7.1 Position Reference

```
Notes{

This program demonstrates the use of the APC with the position reference.

Position Reference
==================
The position setpoint is set in 0.1 motor revolutions, entered into parameter 18.11.

Motion Profile Set up
=====================
The profile speed is 90% of Max speed set for the Drive in parameter 01.06
In this example it is set to 1000rpm, where the profile speed will be set to 900rpm.
The acceleration is 2sec and deceleration is 1sec, to get to 900rpm.

Encoder Resolution
==================
Any encoder feedback can be used with this program, the number of counts per
revolution is 65536.

Drive Enable interlock
======================
When the APC run mode is interlocked with the drive Reference 0n parameter #01.11.
This parameter is set to 1 when the drive is healthy, Enabled and a run is commanded.
It is important to ensure the position loop reference integrators are set to correct
values
If the feedback encoder is moved while in disabled mode.
There are two ways the APC can be set when returning from a disable mode.
Absolute Mode - Start from current position (e.g. track the feedback encoder while
disabled)
Relative Mode - Reset integrator counter to 0 or offset position when disabled

#18.31- Selects absolute position mode.
0 = Position is reset to zero (plus offset if required) position on drive disable
(Relative Mode)
1 = Position is returned to the current position (plus offset if required) on drive
disable (Absolute mode)
When the Drive is re-enabled the position loop reference integrators are set to the
Feedback integrated position.
Both modes works with all encoders

Drive & Motor Setup
===================
Before this program can be used the following must be configured: -
1. Drive run and enable control.
2. Enter motor map data & perform an autotune.
3. Tune the speed loop gains
4. SM-Applications must be in a run state (##.13 = 1)

OTHER NOTES
===========
All the APC parameters have been set up within the initial task, except the position
setpoint.
If changes are required to take effect the SM-Apps will need to be reset.
*** - Denotes default setting, this APC parameter does not need to be set.

} //Notes

Initial{
// Initialisation Program

// DRIVE PARAMETER SETUP
// *********************************************************************
#01.06  = 1000  // Max Speed (1000rpm)
#01.07  = 0     // Min. Speed (0rpm)
#01.10  = 1     // Bipolar Reference
#01.14  = 3     // Preset Speed
```

```
//Reference
#01.15  = 1     // Preset Speed Ref 1
#02.02  = 0     // Disable Drive Ramps
#02.04  = 0     // Fast Decel Ramps
#06.01  = 2     // No Drive Ramp mode

// SM-APPLICATIONS PARAMETER SETUP
// ***********************************************************************
#81.12 = 3  // 1ms Task time
#81.14 = 1  // Global Trip Enable
#81.16 = 1 // Update Ref and Fbck every POS Task
#81.17 = 0  // Disable Parameter Over-range trip
#81.20 = 1  // Save PLC Registers On Power down
REINIT


// INITIALISATION OF VARIABLES
// ***********************************************************************
// None
// APC SETUP
// ***********************************************************************

// Control/Initialise
// =======================================================================
// Disable APC - Track feedback position.
// This will initialise the APC parameters
RunModeStatus% = APCSetRunMode(APC_DISABLE)

// Reference & Feedback Encoders
// =======================================================================
// Counter Reset On Disable ***
DisRstStatus%= APCResetSourcesOnDisable()

// Reset Offset Position (0) ***
OffStatus%= APCSetPositionResetOffset(0)

// Positioning Mode
IF #18.31 = 1 THEN

    // Absolute Mode
    AbsStatus%= APCSelectAbsoluteMode()

ELSE
    // Relative Mode
    RelStatus%= APCSelectRelativeMode()

ENDIF

// Feedback Encoder Source (Drive) ***
FbkSrcStatus%= APCSetFeedbackSource(APC_DRIVE_ENC)

// Feedback Do not Invert ***
FbkInvStatus%= APCDoNotInvertFbckSource()

// Resolution Number of Turn Bits (65536 cpr) ***
NTBStatus%= APCSetNumOfTurnsBits(16)

// Stop Reference
// =======================================================================
// Decelerate on a Stop ***
StopStatus% = APCSetStopMode(APC_PROF_STOP)

// Position Reference
// =======================================================================
// Set Position Reference (Position)
PosSelStatus%= APCSelectReference(APC_POSITION_REF)
// Profile Generator
// =======================================================================
// Enable Profile Generator ***
ProfEnStatus%= APCEnableProfile()
```

```
// Profile Max Speed (90% of 1000rpm = 900rpm)
ProfMaxSpeed% = MULDIV(1073741824, (#01.06 * 90), (60000 * 100))
ProfMxSpStatus%= APCSetProfileMaxSpeedClamp(ProfMaxSpeed%)

// Profile Acceleration and Deceleration Rates (2s-Accel & 1s-Decel)
ProfAcceleration% = MULDIV(ProfMaxSpeed%, 250, 2000000)
ProfDeceleration% = MULDIV(ProfMaxSpeed%, 250, 1000000)
ProfAccStatus1% = APCSetProfileAccelRate(ProfAcceleration%)
ProfDecStatus2% = APCSetProfileDecelRate(ProfDeceleration%)

// Position Loop
// ========================================================================
// Disable external Speed and Position References ***
ExtSpStatus%= APCDisableExternalRefSpeed()
ExtPosStatus%= APCDisableExternalRefPosition()

// Position Loop External Numerator-Denominator (1:1) ***
PNumStatus%= APCSetOutputRatioNumerator(1000)
PDenomStatus%= APCSetOutputRatioDenominator(1000)

// Output Channel (speed ref #01.21)
APCChStatus1%= APCSetupOutputChannel(2, 0, 0)
APCChStatus2%= APCEnableOutputChannel()

// Proportional Term ***
PtermStatus%= APCSetPGain(2500)

// Proportional Term Output Clamp (10% of 1000, head room for position correction)
PGainMaxSpeed%= MULDIV(1073741824, (#01.06 * 10), (60000 * 100))
PTermClStatus%= APCSetPGainSpeedClamp(PGainMaxSpeed%)

// Output Speed Clamp ***
OutClampStatus%= APCSetOutputSpeedClamp(53687091)// equal to 3000rpm

} //Initial

Background{
// Background Task Lowest Priority Task
TOP:

// Position Setpoint Scaling from 0.1revs to counts
PosSetpointCnts% = MULDIV(#18.11, 65536, 10)

// Current Position Indication from counts to 0.1revs
FeedbackPosCnts% = APCReadPar(APC_FB_POS)
#18.12 = MULDIV(FeedbackPosCnts%, 10, 65536)

// Parameter Read Pointer
_P01% = APCReadPar(_P00%)

GOTO TOP:

} //Background

Pos0{
// POS0 Task Runs Before APC Runs

// Enable APC with the Drive Reference ON
IF #01.11 = 1 THEN

    // Enable APC
    RunModeStatus% = APCSetRunMode(APC_ENABLE)

    // Position Setpoint in 0.1 Revolutions of Motor
    PosRefStatus% = APCSetPositionSetPoint(PosSetpointCnts%)

ELSE

    // Disable APC - Track feedback position
    RunModeStatus% = APCSetRunMode(APC_DISABLE)

ENDIF

} //Pos0
```

7.2 Digital Lock - Simple Flying Shear

```
Notes{

This program demonstrates the use of the APC with Digital Lock, and switch to
position reference on the Fly, to simulate a
simple flying shear application.

Position Reference
==================
The position setpoint is set in 0.1 motor revolution, entered into parameter 18.11.

Digital-Lock
============
Parameter #18.32 enables Digital lock.
0 - Position reference is selected and the axis will return to datum set in parameter
#18.11
1 - Enable digital lock. Axis will ramp to line speed and recover any lost position
during acceleration and then lock to the line reference

The master reference is generated as virtual reference within the code to demonstrate
digital lock and also the way how to configure a user
positional reference within a program. The virtual reference is set to 700rpm to
ensure there is enough head room for the feedback axis to
recover the Lost position during acceleration.

Digital lock ration is set to 1:1

Motion Profile Set up
=====================
The profile speed is 90% of Max speed set in Drive #01.06 in this example it is set
to 1000rpm, where the profile speed will be set to 900rpm.
The acceleration is 2sec and deceleration is 1sec, to get to 900rpm.

Encoder Resolution
==================
Any encoder feedback can be use with this program, the number of counts per
revolution is 65536.

Drive Enable interlock
======================
When the APC run mode is interlocked with the drive Reference 0n parameter #01.11.
This parameter is set to 1 when the drive is healthy, Enabled and a run is commanded.
It is important to ensure the position loop reference integrators are set to correct
values
if the feedback encoder is moved while in disabled mode.
There are two ways the APC can be set when returning from a disable mode.
Absolute Mode - Start from current position (e.g. track the feedback encoder while
disabled)
Relative Mode - Reset integrator counter to 0 or offset position when disabled
#18.31- Selects absolute position mode.
0 = Position is reset to zero (plus offset if required) position on drive disable
(Relative Mode)
1 = Position is retained to current position (plus offset if required) on drive
disable (Absolute mode)
When the Drive is re-enabled the position loop reference integrators are set to the
Feedback integrated position.
Ensure the run signal is off before the program is ran in this mode so the reference
integrator can be preset to feedback.

Both modes works with all Unidrive-SP encoders
Drive & Motor Setup
====================
Before this program can be used the following must be configured: -
1. Drive run and enable control.
2. Enter motor map data & perform an autotune.
3. Tune the speed loop gains
4. SM-Applications must be in a run state (##.13 = 1)
```

```
OTHER NOTES
===========
All APC parameters have been set up within the initial task, except the position
setpoint. If changes are required to take effect the
SM-Apps will need to be reset.
*** - Denotes default setting, this APC parameter does not need to be set.

} //Notes


Initial{
// Initialisation Program

// DRIVE PARAMETER SETUP
// **********************************************************************
#01.06  = 1000  // Max Speed (1000rpm)
#01.07  = 0     // Min. Speed (0rpm)
#01.10  = 1     // Bipolar Reference
#01.14  = 3     // Preset Speed Reference
#01.15  = 1     // Preset Speed Ref 1
#02.02  = 0     // Disable Drive Ramps
#02.04  = 0     // Fast Decel Ramps
#06.01  = 2     // No Drive Ramp mode

// SM-APPLICATIONS PARAMETER SETUP
// **********************************************************************
#81.12 = 3  // 1ms Task time
#81.14 = 1  // Global Trip Enable
#81.16 = 1 // Update Ref and Fbck every POS Task
#81.17 = 0  // Disable Parameter Over-range trip
#81.20 = 1  // Save PLC Registers On Power down
REINIT

// INITIALISATION OF VARIABLES
// **********************************************************************
MasterCounter% = 0

// APC SETUP
// **********************************************************************

// Control/Initialise
// ======================================================================
// Disable APC - Track feedback position.
// This will initialise the APC parameters
RunModeStatus% = APCSetRunMode(APC_DISABLE)

// Reference & Feedback Encoders
// ======================================================================
// Counter Reset On Disable ***
DisRstStatus%= APCResetSourcesOnDisable()

// Reset Offset Position (0) ***
OffStatus%= APCSetPositionResetOffset(0)
// Positioning Mode
IF #18.31 = 1 THEN

    // Absolute Mode
    AbsStatus%= APCSelectAbsoluteMode()

ELSE
    // Relative Mode
    RelStatus%= APCSelectRelativeMode()

ENDIF
// Feedback Encoder Source (Drive) ***
FbkSrcStatus%= APCSetFeedbackSource(APC_DRIVE_ENC)

// Feedback Do not Invert ***
FbkInvStatus%= APCDoNotInvertFbckSource()
```

```
// Reference Encoder Source (User)
RefSrcStatus%= APCSetReferenceSource(APC_USER_ENC)

// Reference Do not Invert ***
RefInvStatus%= APCDoNotInvertRefSource()

// Resolution Number of Turn Bits (65536 cpr) ***
NTBStatus%= APCSetNumOfTurnsBits(16)

// Stop Reference
// ======================================================================
// Decelerate on a Stop ***
StopStatus% = APCSetStopMode(APC_PROF_STOP)

// Digital Lock
// ======================================================================
// Digit Lock Mode (ramp and seek lock) ***
DgLkModeStatus% = APCSetDigLockMode(APC_UNLOCKED)

// Digital Lock Type (Rigid)
DgLkRigStatus% = APCEnableRigidLock()

// Digital Lock Ratio ***
DgLkNumStatus% = APCSetDigLockRatioNumerator(1000)
DgLkDenStatus% = APCSetDigLockRatioDenominator(1000)

// Digital Lock Window ***
DgLkWpStatus% = APCSetDigLockLockingPosition(6553)
DgLkWsStatus% = APCSetDigLockLockingSpeed(17895)

// Digital Lock Offsets (0) ***
DgLkOsStatus% = APCSetSpeedOffset(0)
DgLkOpStatus% = APCSetPositionOffset(0)

// Enable Profile Generator ***
ProfEnStatus%= APCEnableProfile()

// Profile Generator
// ======================================================================
// Profile Max Speed (90% of 1000rpm = 900rpm)
ProfMaxSpeed% = MULDIV(1073741824, (#01.06 * 90), (60000 * 100))
ProfMxSpStatus%= APCSetProfileMaxSpeedClamp(ProfMaxSpeed%)

// Profile Acceleration and Deceleration Rates (2s-Accel & 1s-Decel)
ProfAcceleration% = MULDIV(ProfMaxSpeed%, 250, 2000000)
ProfDeceleration% = MULDIV(ProfMaxSpeed%, 250, 1000000)
ProfAccStatus1% = APCSetProfileAccelRate(ProfAcceleration%)
ProfDecStatus2% = APCSetProfileDecelRate(ProfDeceleration%)

// Position Loop
// ======================================================================
// Disable external Speed and Position References ***
ExtSpStatus%= APCDisableExternalRefSpeed()
ExtPosStatus%= APCDisableExternalRefPosition()

// Position Loop External Numerator-Denominator (1:1) ***
PNumStatus%= APCSetOutputRatioNumerator(1000)
PDenomStatus%= APCSetOutputRatioDenominator(1000)

// Output Channel (speed ref #01.21)
APCChStatus1%= APCSetupOutputChannel(2, 0, 0)
APCChStatus2%= APCEnableOutputChannel()

// Proportional Term ***
PtermStatus%= APCSetPGain(2500)

// Proportional Term Output Clamp (10% of 1000, head room for position correction)
PGainMaxSpeed%= MULDIV(1073741824, (#01.06 * 10), (60000 * 100))
PTermClStatus%= APCSetPGainSpeedClamp(PGainMaxSpeed%)

// Output Speed Clamp ***
OutClampStatus%= APCSetOutputSpeedClamp(53687091)// equal to 3000rpm

} //Initial
```

```
Background{
// Background Task Lowest Priority Task

TOP:

// Position Setpoint Scaling from 0.1revs to counts
PosSetpointCnts% = MULDIV(#18.11, 65536, 10)

// Current Position Indication from counts to 0.1revs
FeedbackPosCnts% = APCReadPar(APC_FB_POS)
#18.12 = MULDIV(FeedbackPosCnts%, 10, 65536)

// Parameter Read Pointer
_P01% = APCReadPar(_P00%)

GOTO TOP:

} //Background

Pos0{
// POS0 Task Runs Before APC Runs
// VIRTUAL MASTER COUNTER
// *************************************************************************
// Master Speed = 700rpm
// Counts per POS0 Task sample (1ms) = (RPM * Counts per Rev) / (60 * 1000)
CountsPerSample%= MULDIV(700, 65536, (60000))

// Virtual Counter
MasterCounter%= MasterCounter% + CountsPerSample%

// APC User Program Reference, Dissection of Virtual Counter
Turns%  = UnsignedTopWord(MasterCounter%)
Position% = UnsignedBottomWord(MasterCounter%)
PositionFine% = 0

// Set User Reference
UserRefStatus% = APCSetReferencePosition(Turns%, Position%, PositionFine%)

// DIGITAL LOCK PROGRAM
// *************************************************************************
// Enable APC with the Drive Reference ON
IF #01.11 = 1 THEN

// Enable APC
    RunModeStatus% = APCSetRunMode(APC_ENABLE)
// Enable Digital lock
    IF #18.32 = 1 THEN

// Enable Digital Lock Reference
        PosSelStatus%= APCSelectReference(APC_DIG_LOCK_REF)

// Return to Datum
    ELSE

// Enable Position Reference
        PosSelStatus%= APCSelectReference(APC_POSITION_REF)

// Position Setpoint (0 - datum position)
        PosRefStatus% = APCSetPositionSetPoint(PosSetpointCnts%)

    ENDIF

ELSE

    // Disable APC - Track feedback position
    RunModeStatus% = APCSetRunMode(APC_DISABLE)

    // Set Position Reference (Stop)
    PosSelStatus%= APCSelectReference(APC_STOP_REF)

ENDIF

} //Pos0
```

7.3 CAM

```
Notes{

This program demonstrates the use of the APC with the CAM and its' features (One
shot, Interpolation and dynamic change of CAM start position).

CAM
===
Cam profile generated will be based on the following: -
    V
    ^    _____________ 700rpm(ref & Fbk)
    |   /             \
    |  /               \
    | /                 \
    |/                   \
   -|----|---------|------>t
      2s       5s      3s

Parameter #18.32 enables CAM
0 - Will enable a stop and disable CAM
1 - Enable CAM. This will perform a simple synchronous trapezoidal profile

Parameter #18.33 enables Reverse Profile with Sine Ramps
0 - Forward Profile with linear ramps
1 - Reverse Profile with Sine ramps

Parameter #18.34 enables Single Shot CAM cycle
0 - Cyclic mode
1 - Single Shot Cycle.

The master reference is generated as virtual reference within the code to demonstrate
the CAM and also the way how to configure a user
positional reference within a program. The virtual reference is set to 700rpm to
ensure there is enough head room for the P term trim
to compensate for any position error.

Parameter #18.11 Position reference from CAM output in 0.1 revs
Parameter #18.12 Actual position feedback in 0.1 revs

Motion Profile Set up
=====================
The profile speed is 90% of Max speed set in Drive #01.06 in this example it is set
to 1000rpm, where the profile speed will be set to 900rpm.
The acceleration is 2sec and deceleration is 1sec, to get to 900rpm.
==================
Any encoder feedback can be use with this program, the number of counts per
revolution for the reference and feedback will be 65536.

Drive Enable interlock
======================
When the APC run mode is interlocked with the drive Reference 0n parameter #01.11.
This parameter is set to 1 when the drive is healthy, Enabled and a run is commanded.
It is important to ensure the position loop reference integrators are set to correct
values
if the feedback encoder is moved while in disabled mode.
There are two ways the APC can be set when returning from a disable mode.
Absolute Mode - Start from current position (e.g. track the feedback encoder while
disabled)
Relative Mode - Reset integrator counter to 0 or offset position when disabled
#18.31- Selects absolute position mode.
0 = Position is reset to zero (plus offset if required) position on drive disable
(Relative Mode)
1 = Position is retained to current position (plus offset if required) on drive
disable (Absolute mode)
When the Drive is re-enabled the position loop reference integrators are set to the
Feedback integrated position.
Ensure the run signal is off before the program is ran in this mode so the reference
integrator can be preset to feedback.

Both modes works with all Unidrive-SP encoders
```

```
Drive & Motor Setup
===================
Before this program can be used the following must be configured: -
1. Drive run and enable control.
2. Enter motor map data & perform an autotune.
3. Tune the speed loop gains
4. SM-Applications must be in a run state (##.13 = 1)

OTHER NOTES
===========
All the APC parameters have been set up within the initial task, except the position
setpoint. If changes are required to take effect the
SM-Apps will need to be reset.
*** - Denotes default setting, this APC parameter does not need to be set.

} //Notes


Initial{
// Initialisation Program

// DRIVE PARAMETER SETUP
// **********************************************************************
#01.06  = 1000        // Max Speed (1000rpm)
#01.07  = 0     // Min. Speed (0rpm)
#01.10  = 1     // Bipolar Reference
#01.14  = 3     // Preset Speed Reference
#01.15  = 1     // Preset Speed Ref 1
#02.02  = 0     // Disable Drive Ramps
#02.04  = 0     // Fast Decel Ramps
#06.01  = 2     // No Drive Ramp mode

// SM-APPLICATIONS PARAMETER SETUP
// **********************************************************************
#81.12 = 3  // 1ms Task time
#81.14 = 1  // Global Trip Enable
#81.16 = 1 // Update Ref and Fbck every POS Task
#81.17 = 0 // Disable Parameter Over-range trip
#81.20 = 1 // Save PLC Registers On Power down
REINIT

// INITIALISATION OF VARIABLES
// **********************************************************************
MasterCounter% = 0

// DIMESIONING CAM ARRAYS
// **********************************************************************
DIM InArray%[6]
DIM OutArray%[6]
DIM IntArray%[6]

// CAM ARRAY ELEMENTS
```

```
// ********************************************************************
//
//  V
//  V
//  ^    _____________ 700rpm(ref & Fbk)
//  |   /              \
//  |  /                \
//  | /                  \
//  |/                    \
// -|----|---------|------>t
//    2s       5s      3s
//
// COUNTS PER REV
// Reference = 65536
// Feedback  = 65536
//
// REFERENCE ARRAY
// Reference Speed CPS = 65536 * 700 / 60 = 764587
InArray%[0]= (764587 * 2)// (2sec) Forward direction
InArray%[1]= (764587 * 5)// (5sec)
InArray%[2]= (764587 * 3)// (3sec)
InArray%[3]= (764587 * 2)// (2sec)Reverse direction
InArray%[4]= (764587 * 5)// (5sec)
InArray%[5]= (764587 * 3)// (3sec)

// FEEDBACK ARRAY
// Feedback Speed = Reference speed = 764587
// Accel/Decel Distance = Vel/2*Time
// Steady State Distance = Vel/Time
OutArray%[0]=   MULDIV(764587, 2, 2)// Forward direction
OutArray%[1]=   MULDIV(764587, 5, 1)
OutArray%[2]=   MULDIV(764587, 3, 2)
OutArray%[3]= -(MULDIV(764587, 2, 2))// Reverse direction
OutArray%[4]= -(MULDIV(764587, 5, 1))
OutArray%[5]= -(MULDIV(764587, 3, 2))

// INTERPOLATION ARRAY
IntArray%[0]=   2// Forward direction (linear Ramps)
IntArray%[1]=   0
IntArray%[2]=   3
IntArray%[3]=   4// Reverse direction (Sine Ramps)
IntArray%[4]=   0
IntArray%[5]=   5

// APC SETUP
// ********************************************************************
// Control/Initialise
// ====================================================================
// Disable APC - Track feedback position.
// This will initialise the APC parameters
RunModeStatus% = APCSetRunMode(APC_DISABLE)

// Reference & Feedback Encoders
// ====================================================================
// Counter Reset On Disable ***
DisRstStatus%= APCResetSourcesOnDisable()

// Reset Offset Position (0) ***
OffStatus%= APCSetPositionResetOffset(0)

// Positioning Mode
IF #18.31 = 1 THEN

    // Absolute Mode
    AbsStatus%= APCSelectAbsoluteMode()

ELSE
    // Relative Mode
    RelStatus%= APCSelectRelativeMode()

ENDIF
```

```
// Feedback Encoder Source (Drive) ***
FbkSrcStatus%= APCSetFeedbackSource(APC_DRIVE_ENC)

// Feedback Do not Invert ***
FbkInvStatus%= APCDoNotInvertFbckSource()

// Reference Encoder Source (User)
RefSrcStatus%= APCSetReferenceSource(APC_USER_ENC)

// Reference Do not Invert ***
RefInvStatus%= APCDoNotInvertRefSource()

// Resolution Number of Turn Bits (65536 cpr) ***
NTBStatus%= APCSetNumOfTurnsBits(16)

// Stop Reference
// =========================================================================
// Decelerate on a Stop ***
StopStatus% = APCSetStopMode(APC_PROF_STOP)

// CAM Reference
// =========================================================================
// Initialise CAM2 function with In, Out & Interpolation Array
CAMIniStatus% = APCCamInitialise2(InArray%, OutArray%, IntArray%)

// Interpolation Type (Array)
CAMIntStatus% = APCSetCAMInterpolationMode(APC_ARRAY_CAM)

// CAM Segment limit ***
CAMsegStatus% = APCSetCAMDeltaSegLimit(256)

// CAM Output Ratio ***
CAMNumStatus% = APCSetCAMOutRatioNumerator(1000)
CAMNumStatus% = APCSetCAMOutRatioDenominator(1000)

// Start Point and Size
CAMStrStatus% = APCSetCAMStartIndex(0)
CAMSzStatus% = APCSetCAMSize(3)

// CAM Start Mode
CAMStrStatus% = APCSelectCAMZeroReset()
// CAMStrStatus% = APCSelectCAMAbsoluteReset()
// IndexStatus% = APCSetCAMAbsResetIndex(Index%)
// SegmentStatus% = APCSetCAMAbsResetPositionInSeg(Position%)

// Profile Generator
// =========================================================================
// Enable Profile Generator ***
ProfEnStatus%= APCEnableProfile()

// Profile Max Speed (90% of 1000rpm = 900rpm)
ProfMaxSpeed% = MULDIV(1073741824, (#01.06 * 90), (60000 * 100))
ProfMxSpStatus%= APCSetProfileMaxSpeedClamp(ProfMaxSpeed%)

// Profile Acceleration and Deceleration Rates (2s-Accel & 1s-Decel)
ProfAcceleration% = MULDIV(ProfMaxSpeed%, 250, 2000000)
ProfDeceleration% = MULDIV(ProfMaxSpeed%, 250, 1000000)
ProfAccStatus1% = APCSetProfileAccelRate(ProfAcceleration%)
ProfDecStatus2% = APCSetProfileDecelRate(ProfDeceleration%)

// Position Loop
// =========================================================================
// Disable external Speed and Position References ***
ExtSpStatus%= APCDisableExternalRefSpeed()
ExtPosStatus%= APCDisableExternalRefPosition()

// Position Loop External Numerator-Denominator (1:1) ***
PNumStatus%= APCSetOutputRatioNumerator(1000)
PDenomStatus%= APCSetOutputRatioDenominator(1000)

// Output Channel (speed ref #01.21)
APCChStatus1%= APCSetupOutputChannel(2, 0, 0)
APCChStatus2%= APCEnableOutputChannel()
```

```
// Proportional Term ***
PtermStatus%= APCSetPGain(2500)

// Proportional Term Output Clamp (10% of 1000, head room for position correction)
PGainMaxSpeed%= MULDIV(1073741824, (#01.06 * 10), (60000 * 100))
PTermClStatus%= APCSetPGainSpeedClamp(PGainMaxSpeed%)

// Output Speed Clamp ***
OutClampStatus%= APCSetOutputSpeedClamp(53687091)// equal to 3000rpm

} //Initial


Background{
// Background Task Lowest Priority Task

TOP:

// Current Position Indication (0.1rev)
PosRefCnts% = APCReadPar(APC_PROF_IN_POS)
#18.11 = MULDIV(PosRefCnts%, 10, 65536)

// Current Reference Position Indication (0.1rev)
FeedbackPosCnts% = APCReadPar(APC_FB_POS)
#18.12 = MULDIV(FeedbackPosCnts%, 10, 65536)

// Parameter Read Pointer
_P01% = APCReadPar(_P00%)

GOTO TOP:

} //Background


Pos0{
// POS0 Task Runs Before APC Runs

// VIRTUAL MASTER COUNTER
// *************************************************************************
// Master Speed = 700rpm
// Counts per POS0 Task sample (1ms) = (RPM * Counts per Rev) / (60 * 1000)
CountsPerSample%= MULDIV(700, 65536, (60000))

// Virtual Counter
MasterCounter%= MasterCounter% + CountsPerSample%

// APC User Program Reference, Dissection of Virtual Counter
Turns%  = UnsignedTopWord(MasterCounter%)
Position% = UnsignedBottomWord(MasterCounter%)
PositionFine% = 0

// Set User Reference
UserRefStatus% = APCSetReferencePosition(Turns%, Position%, PositionFine%)
```

```
// CAM PROGRAM
// ***********************************************************************

// Enable APC with the Drive Reference ON
IF #01.11 = 1 THEN

    // Enable APC
    RunModeStatus% = APCSetRunMode(APC_ENABLE)

    // Enable CAM
    IF #18.32 = 1 THEN

        // Enable CAM Reference
        PosSelStatus%= APCSelectReference(APC_CAM_REF)

        // Forward CAM Array Elements
        IF #18.33 = 1 THEN

        CAMStrStatus% = APCSetCAMStartIndex(3)
        CAMSzStatus% = APCSetCAMSize(3)

        // Reverse CAM Array Elements
        ELSE

        CAMStrStatus% = APCSetCAMStartIndex(0)
        CAMSzStatus% = APCSetCAMSize(3)

        ENDIF

        // Single Shot
        IF #18.34 = 1 THEN

        CAMSSStatus% = APCEnableCAMSingleShot()

        // Cyclic
        ELSE

        CAMSSStatus% = APCDisableCAMSingleShot()

        ENDIF

    // Decelerate to a Stop
    ELSE

        // Enable Position Reference
        PosSelStatus%= APCSelectReference(APC_STOP_REF)


    ENDIF

ELSE

    // Disable APC - Track feedback position
    RunModeStatus% = APCSetRunMode(APC_DISABLE)

    // Set Position Reference (Stop)
    PosSelStatus%= APCSelectReference(APC_STOP_REF)

ENDIF

} //Pos0
```

7.4 Speed and Position - Homing

```
Notes{

This program demonstrates the use of the APC using the Speed and Position References
with the Marker Pulse Capture, to perform a simple home on marker function.

Homing Description
==================
With the Unidrive in run mode, when parameter #18.32 is set to 1 (positive edge
triggered), the motor will rotate at 10rpm in speed mode.
When the motor encoder marker is detected the APC is switched to Position reference
and the position reference is set to the capture marker position.
When the marker position is achieved with +/-20 counts then home is completed.


Motion Profile Set up
=====================
The profile speed is 90% of Max speed set in Drive #01.06
In this example it is set to 1000rpm, where the profile speed will be set to 900rpm.
The acceleration is 2sec and deceleration is 1sec, to get to 900rpm.

Encoder Resolution
==================
Any encoder feedback can be use with this program, the number of counts per
revolution is 65536.

Drive Enable interlock
======================
When the APC run mode is interlocked with the drive Reference 0n parameter #01.11.
This parameter is set to 1 when the drive is healthy, Enabled and a run is commanded.
It is important to ensure the position loop reference integrators are set to correct
values
if the feedback encoder is moved while in disabled mode.
There are two ways the APC can be set when returning from a disable mode.
Absolute Mode - Start from current position (e.g. track the feedback encoder while
disabled)
Relative Mode - Reset integrator counter to 0 or offset position when disabled

#18.31- Selects absolute position mode.
          0 = Position is reset to zero (plus offset if required) position on drive
disable (Relative Mode)
          1 = Position is retained to current position (plus offset if required) on
drive disable (Absolute mode)
              When the Drive is re-enabled the position loop reference integrators
are set to the Feedback integrated position.
Both modes works with all encoders

Drive & Motor Setup
===================
Before this program can be used the following must be configured: -
1. Drive run and enable control.
2. Enter motor map data & perform an autotune.
3. Tune the speed loop gains
4. SM-Applications must be in a run state (##.13 = 1)

OTHER NOTES
===========
All the APC parameters have been set up within the initial task, except the position
setpoint.
If changes are required to take effect the SM-Apps will need to be reset.
*** - Denotes default setting, this APC parameter does not need to be set.

} //Notes

Initial{
// Initialisation Program

// DRIVE PARAMETER SETUP
// *********************************************************************
#01.06  = 1000  // Max Speed (1000rpm)
#01.07  = 0     // Min. Speed (0rpm)
#01.10  = 1     // Bipolar Reference
#01.14  = 3     // Preset Speed Reference
```

```
#01.15  = 1     // Preset Speed Ref 1
#02.02  = 0     // Disable Drive Ramps
#02.04  = 0     // Fast Decel Ramps
#06.01  = 2     // No Drive Ramp mode

// SM-APPLICATIONS PARAMETER SETUP
// *************************************************************************
#81.12 = 3  // 1ms Task time
#81.14 = 1  // Global Trip Enable
#81.16 = 1 // Update Ref and Fbck every POS Task
#81.17 = 0  // Disable Parameter Over-range trip
#81.20 = 1  // Save PLC Registers On Power down
REINIT

// INITIALISATION OF VARIABLES
// *************************************************************************
// None

// APC SETUP
// *************************************************************************
Homing%     = 0
Old1832%    = 0
OldMarkerFlag%= 0
FeedbackPositionCnts%= 0

// Control/Initialise
// =========================================================================
// Disable APC - Track feedback position.
// This will initialise the APC parameters
RunModeStatus% = APCSetRunMode(APC_DISABLE)

// Reference & Feedback Encoders
// =========================================================================
// Counter Reset On Disable ***
DisRstStatus%= APCResetSourcesOnDisable()

// Reset Offset Position (0) ***
OffStatus%= APCSetPositionResetOffset(0)

// Positioning Mode
IF #18.31 = 1 THEN

    // Absolute Mode
    AbsStatus%= APCSelectAbsoluteMode()

ELSE
    // Relative Mode
    RelStatus%= APCSelectRelativeMode()

ENDIF

// Feedback Encoder Source (Drive) ***
FbkSrcStatus%= APCSetFeedbackSource(APC_DRIVE_ENC)

// Feedback Do not Invert ***
FbkInvStatus%= APCDoNotInvertFbckSource()
// Do not reset feedback position on first Marker ***
DisMrkStatus% = APCDisableFbckSourceMarker()

// Enable Marker Pulse Capture Function
MarkerEnStatus% = APCEnableFeedbackMarker()

// Resolution Number of Turn Bits (65536 cpr) ***
NTBStatus%= APCSetNumOfTurnsBits(16)
```

```
// Position Reference
// ======================================================================
// Set Position Reference (Stop)
PosSelStatus%= APCSelectReference(APC_STOP_REF)

// Stop Type
StpTypStatus%= APCSetStopMode(APC_PROF_STOP)

// Profile Generator
// ======================================================================
// Enable Profile Generator ***
ProfEnStatus%= APCEnableProfile()

// Profile Max Speed (90% of 1000rpm = 900rpm)
ProfMaxSpeed% = MULDIV(1073741824, (#01.06 * 90), (60000 * 100))
ProfMxSpStatus%= APCSetProfileMaxSpeedClamp(ProfMaxSpeed%)

// Profile Acceleration and Deceleration Rates (2s-Accel & 1s-Decel)
ProfAcceleration% = MULDIV(ProfMaxSpeed%, 250, 2000000)
ProfDeceleration% = MULDIV(ProfMaxSpeed%, 250, 1000000)
ProfAccStatus1% = APCSetProfileAccelRate(ProfAcceleration%)
ProfDecStatus2% = APCSetProfileDecelRate(ProfDeceleration%)

// Position Loop
// ======================================================================
// Disable external Speed and Position References ***
ExtSpStatus%= APCDisableExternalRefSpeed()
ExtPosStatus%= APCDisableExternalRefPosition()

// Position Loop External Numerator-Denominator (1:1) ***
PNumStatus%= APCSetOutputRatioNumerator(1000)
PDenomStatus%= APCSetOutputRatioDenominator(1000)

// Output Channel (speed ref #01.21)
APCChStatus1%= APCSetupOutputChannel(2, 0, 0)
APCChStatus2%= APCEnableOutputChannel()
// Proportional Term ***
PtermStatus%= APCSetPGain(2500)

// Proportional Term Output Clamp (10% of 1000, head room for position correction)
PGainMaxSpeed%= MULDIV(1073741824, (#01.06 * 10), (60000 * 100))
PTermClStatus%= APCSetPGainSpeedClamp(PGainMaxSpeed%)

// Output Speed Clamp ***
OutClampStatus%= APCSetOutputSpeedClamp(53687091)// equal to 3000rpm

} //Initial


Background{
// Background Task Lowest Priority Task

TOP:

// Current Position Indication (0.1rev)
PosRefCnts% = APCReadPar(APC_PROF_IN_POS)
#18.11 = MULDIV(PosRefCnts%, 10, 65536)

// Current Reference Position Indication (0.1rev)
FeedbackPosCnts% = APCReadPar(APC_FB_POS)
#18.12 = MULDIV(FeedbackPosCnts%, 10, 65536)

// Parameter Read Pointer
_P01% = APCReadPar(_P00%)

IF #18.32 > Old1832% THEN Homing% = 1

Old1832% = #18.32


GOTO TOP:

} //Background
```

```
Pos0{
// POS0 Task Runs Before APC Runs

// Read APC Parameters
FeedbackPositionCnts% = APCReadPar(APC_FB_POS)
MarkerFlag% = APCReadPar(APC_FBSRC_MRKFLG)

// Enable APC with the Drive Reference ON
IF #01.11 = 1 THEN

    // Enable APC
    RunModeStatus% = APCSetRunMode(APC_ENABLE)

    // Home - Set Sped Look For Marker
    IF Homing% = 1 THEN

        // Reset Marker flag, arm for next capture
        MrkRstStatus% = APCResetFbckSourceMarkerFlag()

        // Select Speed Reference
        PosSelStatus%= APCSelectReference(APC_SPEED_REF)

        // Set Speed Reference (10rpm)
        HomeSpeed%= MULDIV(1073741824, 10, 60000)
        HmSpStatus%= APCSetSpeedSetPoint(HomeSpeed%)

        // Set Homing flag
        Homing% = 2

    // Detect Marker
    ELSEIF Homing% = 2 THEN

        // Check Marker Flag
        IF MarkerFlag% > OldMarkerFlag% THEN Homing% = 3

    // Goto Marker Position
    // Allow one scan of the APC to set position integrators
    ELSEIF Homing% = 3 THEN

        // Read Marker Position
        MarkerPositionCnts% = APCReadPar(APC_FBSRC_MRKPOS)

        // Select Position Reference
         PosSelStatus%= APCSelectReference(APC_POSITION_REF)

        // Set Position Reference to Marker Captured Position
        PosRefStatus% = APCSetPositionSetPoint(MarkerPositionCnts%)

        Homing% = 4

    // Detect When at Marker Position
    ELSEIF Homing% = 4 THEN

        // AT Position Window +/- 20 Counts
        InPosition% = IWINDOW(FeedbackPositionCnts%, MarkerPositionCnts%, 20, 20)

        IF InPosition% = 1 THEN

        // Select Stop Position
PosSelStatus%= APCSelectReference(APC_STOP_REF)

        Homing% = 5

        ENDIF

    ENDIF

ELSE
```

```
// Disable APC - Track feedback position
    RunModeStatus% = APCSetRunMode(APC_DISABLE)

ENDIF

// Cache old states for edge triggering
OldMarkerFlag%= MarkerFlag%

} //Pos0
```

7.5 CTSync Master and Digital Lock

```
Notes{

This program demonstrates the use of the APC using CTSync and direct digital lock on
a Feedback (slave) axis.
This program also generates a Virtual master position, for the remote and local slave
axes.

Program Description
==================
With the drive in Run mode and a bit 0 of the Virtual master control bit 0 set high,
the drive will follow the virtual master reference in rigid digital
-lock, (ramp up and retain position when it was enabled).
The Virtual Master reference is generated also within this program, though the APC is
treated as the feedback (slave), motion axis. The virtual position
is based on the a steady count value which equates to 700rpm.

Motion Profile Set up
=====================
The profile speed is 90% of Max speed set in Drive #01.06
In this example it is set to 1000rpm, where the profile speed will be set to 900rpm.
The acceleration is 2sec and deceleration is 1sec, to get to 900rpm.

Encoder Resolution
==================
Any encoder feedback can be use with this program, the number of counts per
revolution is 65536.

Drive Enable interlock
======================
When the APC run mode is interlocked with the drive Reference 0n parameter #01.11.
This parameter is set to 1 when the drive is healthy, Enabled and a run is commanded.
It is important to ensure the position loop reference integrators are set to correct
values
if the feedback encoder is moved while in disabled mode.
There are two ways the APC can be set when returning from a disable mode.
Absolute Mode - Start from current position (e.g. track the feedback encoder while
disabled)
Relative Mode - Reset integrator counter to 0 or offset position when disabled

#18.31- Selects absolute position mode.
          0 = Position is reset to zero (plus offset if required) position on drive
disable (Relative Mode)
          1 = Position is retained to current position (plus offset if required) on
drive disable (Absolute mode)
              When the Drive is re-enabled the position loop reference integrators
are set to the Feedback integrated position.
Both modes works with all encoders

Drive & Motor Setup
====================
Before this program can be used the following must be configured: -
1. Drive run and enable control.
2. Enter motor map data & perform an autotune.
3. Tune the speed loop gains
4. SM-Applications must be in a run state (##.13 = 1)

OTHER NOTES
===========
All the APC parameters have been set up within the initial task, except the position
setpoint.
If changes are required to take effect the SM-Apps will need to be reset.
*** - Denotes default setting, this APC parameter does not need to be set.

} //Notes
```

```
Initial{
// Initialisation Program

// DRIVE PARAMETER SETUP
// **********************************************************************
#01.06  = 1000            // Max Speed (1000rpm)
#01.07  = 0     // Min. Speed (0rpm)
#01.10  = 1     // Bipolar Reference
#01.14  = 3     // Preset Speed Reference
#01.15  = 1     // Preset Speed Ref 1
#02.02  = 0     // Disable Drive Ramps
#02.04  = 0     // Fast Decel Ramps
#06.01  = 2     // No Drive Ramp mode

// SM-APPLICATIONS PARAMETER SETUP
// **********************************************************************
#81.06 = 25 // CTSync Master
#81.12 = 3  // 1ms Task time
#81.14 = 1  // Global Trip Enable
#81.16 = 1 // Update Ref and Fbck every POS Task
#81.17 = 0  // Disable Parameter Over-range trip
#81.20 = 1  // Save PLC Registers On Power down
REINIT

// INITIALISATION OF VARIABLES
// **********************************************************************
// None
MasterCounter%= 0
#18.32      = 0

// APC SETUP
// **********************************************************************

// Control/Initialise
// ======================================================================
// Disable APC - Track feedback position.
// This will initialise the APC parameters
RunModeStatus% = APCSetRunMode(APC_DISABLE)

// Reference & Feedback Encoders
// ======================================================================
// Counter Reset On Disable ***
DisRstStatus%= APCResetSourcesOnDisable()

// Reset Offset Position (0) ***
OffStatus%= APCSetPositionResetOffset(0)

// Positioning Mode
IF #18.31 = 1 THEN

    // Absolute Mode
    AbsStatus%= APCSelectAbsoluteMode()

ELSE
    // Relative Mode
    RelStatus%= APCSelectRelativeMode()

ENDIF

// Feedback Encoder Source (Drive) ***
FbkSrcStatus%= APCSetFeedbackSource(APC_DRIVE_ENC)

// Feedback Do not Invert ***
FbkInvStatus%= APCDoNotInvertFbckSource()

// Reference Encoder Source (User)
RefSrcStatus%= APCSetReferenceSource(APC_USER_ENC)

// Reference Do not Invert ***
RefInvStatus%= APCDoNotInvertRefSource()
```

```
// Do not reset feedback position on first Marker ***
DisMrkStatus% = APCDisableFbckSourceMarker()

// Resolution Number of Turn Bits (65536 cpr) ***
NTBStatus%= APCSetNumOfTurnsBits(16)

// Stop Reference
// ====================================================================
// Set Position Reference (Stop)
PosSelStatus%= APCSelectReference(APC_STOP_REF)

// Stop Type
StpTypStatus%= APCSetStopMode(APC_PROF_STOP)

// Digital Lock
// ====================================================================
// Digit Lock Mode (ramp and seek lock) ***
DgLkModeStatus% = APCSetDigLockMode(APC_UNLOCKED)

// Digital Lock Type (Ridged)
DgLkRigStatus% = APCEnableRigidLock()

// Digital Lock Ratio ***
DgLkNumStatus% = APCSetDigLockRatioNumerator(1000)
DgLkDenStatus% = APCSetDigLockRatioDenominator(1000)

// Digital Lock Window ***
DgLkWpStatus% = APCSetDigLockLockingPosition(6553)
DgLkWsStatus% = APCSetDigLockLockingSpeed(17895)

// Digital Lock Offsets (0) ***
DgLkOsStatus% = APCSetSpeedOffset(0)
DgLkOpStatus% = APCSetPositionOffset(0)

// Enable Profile Generator ***
ProfEnStatus%= APCEnableProfile()

// Profile Generator
// ====================================================================
// Enable Profile Generator ***
ProfEnStatus%= APCEnableProfile()

// Profile Max Speed (90% of 1000rpm = 900rpm)
ProfMaxSpeed% = MULDIV(1073741824, (#01.06 * 90), (60000 * 100))
ProfMxSpStatus%= APCSetProfileMaxSpeedClamp(ProfMaxSpeed%)

// Profile Acceleration and Deceleration Rates (2s-Accel & 1s-Decel)
ProfAcceleration% = MULDIV(ProfMaxSpeed%, 250, 2000000)
ProfDeceleration% = MULDIV(ProfMaxSpeed%, 250, 1000000)
ProfAccStatus1% = APCSetProfileAccelRate(ProfAcceleration%)
ProfDecStatus2% = APCSetProfileDecelRate(ProfDeceleration%)

// Position Loop
// ====================================================================
// Disable external Speed and Position References ***
ExtSpStatus%= APCDisableExternalRefSpeed()
ExtPosStatus%= APCDisableExternalRefPosition()

// Position Loop External Numerator-Denominator (1:1) ***
PNumStatus%= APCSetOutputRatioNumerator(1000)
PDenomStatus%= APCSetOutputRatioDenominator(1000)

// Output Channel (speed ref #01.21)
APCChStatus1%= APCSetupOutputChannel(2, 0, 0)
APCChStatus2%= APCEnableOutputChannel()

// Proportional Term ***
PtermStatus%= APCSetPGain(2500)

// Proportional Term Output Clamp (10% of 1000, head room for position correction)
PGainMaxSpeed%= MULDIV(1073741824, (#01.06 * 10), (60000 * 100))
PTermClStatus%= APCSetPGainSpeedClamp(PGainMaxSpeed%)
```

```
// Output Speed Clamp ***
OutClampStatus%= APCSetOutputSpeedClamp(53687091)// equal to 3000rpm

} //Initial

Background{
// Background Task Lowest Priority Task

TOP:

// Current Position Indication (0.1rev)
PosRefCnts% = APCReadPar(APC_PROF_IN_POS)
#18.11 = MULDIV(PosRefCnts%, 10, 65536)

// Current Reference Position Indication (0.1rev)
FeedbackPosCnts% = APCReadPar(APC_FB_POS)
#18.12 = MULDIV(FeedbackPosCnts%, 10, 65536)

// Parameter Read Pointer
_P01%      = APCReadPar(_P00%)

GOTO TOP:

} //Background


Pos0{
// POS0 Task Runs Before APC Runs

// SET REFERENCE POSITION FROM CTSync
// **********************************************************************
// Get Virtual Master Reference from CTSync (Only using 32bit Position Counter)
(MasterReference1%, MasterReference2%, MasterControlWord%, Status%) =
CTSyncGetSlaveReferences()

// APC User Program Reference, Dissection of Virtual Counter
Turns%  = UnsignedTopWord(MasterReference1%)
Position% = UnsignedBottomWord(MasterReference1%)
PositionFine% = 0

// Set User Reference
UserRefStatus% = APCSetReferencePosition(Turns%, Position%, PositionFine%)

// MAIN PROGRAM
// **********************************************************************
// Enable APC with the Drive Reference ON
IF #01.11 = 1 THEN

    // Enable APC
    RunModeStatus% = APCSetRunMode(APC_ENABLE)

    IF MasterControlWord% = 1 THEN

        // Enable Digital Lock Reference
        PosSelStatus%= APCSelectReference(APC_DIG_LOCK_REF)

    ELSE
        // Set Position Reference (Stop)
        PosSelStatus%= APCSelectReference(APC_STOP_REF)
ENDIF

ELSE

    // Disable APC - Track feedback position
    RunModeStatus% = APCSetRunMode(APC_DISABLE)

    // Set Position Reference (Stop)
    PosSelStatus%= APCSelectReference(APC_STOP_REF)

ENDIF
```

```
// VIRTUAL MASTER COUNTER
// ***********************************************************************
// Master Speed = 700rpm
// Counts per POS0 Task sample (1ms) = (RPM * Counts per Rev) / (60 * 1000)
CountsPerSample%= MULDIV(700, 65536, (60000))

// Virtual Counter
MasterCounter%= MasterCounter% + CountsPerSample%

// Common Slave Enable
IF #18.32 = 1 THEN

    MasterControl%.0 = 1

ELSE

    MasterControl%.0 = 0

ENDIF

// Set CTSync Virtual Master Reference
CTSyncSetMasterReferences(MasterCounter%, 0, MasterControl%)

} //Pos0
```

7.6 CTSync Slave and Digital Lock

```
Notes{

This program demonstrates the use of the APC using CTSync and direct digital lock on
a Feedback(slave) axis.

Program Description
==================
With the drive in Run mode and a bit 0 of the Virtual master control bit 0 set high,
the drive will follow the virtual master reference in rigid digital
-lock, (ramp up and retain position when it was enabled).

Motion Profile Set up
=====================
The profile speed is 90% of Max speed set in Drive #01.06
In this example it is set to 1000rpm, where the profile speed will be set to 900rpm.
The acceleration is 2sec and deceleration is 1sec, to get to 900rpm.

Encoder Resolution
==================
Any encoder feedback can be use with this program, the number of counts per
revolution is 65536.

Drive Enable interlock
======================
When the APC run mode is interlocked with the drive Reference 0n parameter #01.11.
This parameter is set to 1 when the drive is healthy, Enabled and a run is commanded.
It is important to ensure the position loop reference integrators are set to correct
values
if the feedback encoder is moved while in disabled mode.
There are two ways the APC can be set when returning from a disable mode.
Absolute Mode - Start from current position (e.g. track the feedback encoder while
disabled)
Relative Mode - Reset integrator counter to 0 or offset position when disabled

#18.31- Selects absolute position mode.
          0 = Position is reset to zero (plus offset if required) position on drive
disable (Relative Mode)
          1 = Position is retained to current position (plus offset if required) on
drive disable (Absolute mode)
              When the Drive is re-enabled the position loop reference integrators
are set to the Feedback integrated position.
Both modes works with all encoders

Drive & Motor Setup
====================
Before this program can be used the following must be configured: -
1. Drive run and enable control.
2. Enter motor map data & perform an autotune.
3. Tune the speed loop gains
4. SM-Applications must be in a run state (##.13 = 1)

OTHER NOTES
===========
All the APC parameters have been set up within the initial task, except the position
setpoint.
If changes are required to take effect the SM-Apps will need to be reset.
*** - Denotes default setting, this APC parameter does not need to be set.

} //Notes


Initial{
// Initialisation Program

// DRIVE PARAMETER SETUP
// *********************************************************************
#01.06  = 1000  // Max Speed (1000rpm)
#01.07  = 0     // Min. Speed (0rpm)
#01.10  = 1     // Bipolar Reference
#01.14  = 3     // Preset Speed Reference
```

```
#01.15  = 1     // Preset Speed Ref 1
#02.02  = 0     // Disable Drive Ramps
#02.04  = 0     // Fast Decel Ramps
#06.01  = 2     // No Drive Ramp mode

// SM-APPLICATIONS PARAMETER SETUP
// ***********************************************************************
#81.06 = 26 // CTSync Slave
#81.12 = 3  // 1ms Task time
#81.14 = 1  // Global Trip Enable
#81.16 = 1 // Update Ref and Fbck every POS Task
#81.17 = 0  // Disable Parameter Over-range trip
#81.20 = 1  // Save PLC Registers On Power down
REINIT

// INITIALISATION OF VARIABLES
// ***********************************************************************
// None

// APC SETUP
// ***********************************************************************
// None

// Control/Initialise
// =======================================================================
// Disable APC - Track feedback position.
// This will initialise the APC parameters
RunModeStatus% = APCSetRunMode(APC_DISABLE)

// Reference & Feedback Encoders
// =======================================================================
// Counter Reset On Disable ***
DisRstStatus%= APCResetSourcesOnDisable()

// Reset Offset Position (0) ***
OffStatus%= APCSetPositionResetOffset(0)

// Positioning Mode
IF #18.31 = 1 THEN

    // Absolute Mode
    AbsStatus%= APCSelectAbsoluteMode()

ELSE
    // Relative Mode
    RelStatus%= APCSelectRelativeMode()

ENDIF

// Feedback Encoder Source (Drive) ***
FbkSrcStatus%= APCSetFeedbackSource(APC_DRIVE_ENC)

// Feedback Do not Invert ***
FbkInvStatus%= APCDoNotInvertFbckSource()

// Reference Encoder Source (User)
RefSrcStatus%= APCSetReferenceSource(APC_USER_ENC)

// Reference Do not Invert ***
RefInvStatus%= APCDoNotInvertRefSource()

// Do not reset feedback position on first Marker ***
DisMrkStatus% = APCDisableFbckSourceMarker()

// Resolution Number of Turn Bits (65536 cpr) ***
NTBStatus%= APCSetNumOfTurnsBits(16)

// Stop Reference
```

```
// =====================================================================
// Set Position Reference (Stop)
PosSelStatus%= APCSelectReference(APC_STOP_REF)

// Stop Type
StpTypStatus%= APCSetStopMode(APC_PROF_STOP)

// Digital Lock
// =====================================================================
// Digit Lock Mode (ramp and seek lock) ***
DgLkModeStatus% = APCSetDigLockMode(APC_UNLOCKED)

// Digital Lock Type (Ridged)
DgLkRigStatus% = APCEnableRigidLock()

// Digital Lock Ratio ***
DgLkNumStatus% = APCSetDigLockRatioNumerator(1000)
DgLkDenStatus% = APCSetDigLockRatioDenominator(1000)

// Digital Lock Window ***
DgLkWpStatus% = APCSetDigLockLockingPosition(6553)
DgLkWsStatus% = APCSetDigLockLockingSpeed(17895)

// Digital Lock Offsets (0) ***
DgLkOsStatus% = APCSetSpeedOffset(0)
DgLkOpStatus% = APCSetPositionOffset(0)

// Enable Profile Generator ***
ProfEnStatus%= APCEnableProfile()

// Profile Generator
// =====================================================================
// Enable Profile Generator ***
ProfEnStatus%= APCEnableProfile()
// Profile Max Speed (90% of 1000rpm = 900rpm)
ProfMaxSpeed% = MULDIV(1073741824, (#01.06 * 90), (60000 * 100))
ProfMxSpStatus%= APCSetProfileMaxSpeedClamp(ProfMaxSpeed%)

// Profile Acceleration and Deceleration Rates (2s-Accel & 1s-Decel)
ProfAcceleration% = MULDIV(ProfMaxSpeed%, 250, 2000000)
ProfDeceleration% = MULDIV(ProfMaxSpeed%, 250, 1000000)
ProfAccStatus1% = APCSetProfileAccelRate(ProfAcceleration%)
ProfDecStatus2% = APCSetProfileDecelRate(ProfDeceleration%)

// Position Loop
// =====================================================================
// Disable external Speed and Position References ***
ExtSpStatus%= APCDisableExternalRefSpeed()
ExtPosStatus%= APCDisableExternalRefPosition()

// Position Loop External Numerator-Denominator (1:1) ***
PNumStatus%= APCSetOutputRatioNumerator(1000)
PDenomStatus%= APCSetOutputRatioDenominator(1000)

// Output Channel (speed ref #01.21)
APCChStatus1%= APCSetupOutputChannel(2, 0, 0)
APCChStatus2%= APCEnableOutputChannel()

// Proportional Term ***
PtermStatus%= APCSetPGain(2500)

// Proportional Term Output Clamp (10% of 1000, head room for position correction)
PGainMaxSpeed%= MULDIV(1073741824, (#01.06 * 10), (60000 * 100))
PTermClStatus%= APCSetPGainSpeedClamp(PGainMaxSpeed%)

// Output Speed Clamp ***
OutClampStatus%= APCSetOutputSpeedClamp(53687091)// equal to 3000rpm

} //Initial
```

```
Background{
// Background Task Lowest Priority Task

TOP:

// Current Position Indication (0.1rev)
PosRefCnts% = APCReadPar(APC_PROF_IN_POS)
#18.11 = MULDIV(PosRefCnts%, 10, 65536)

// Current Reference Position Indication (0.1rev)
FeedbackPosCnts% = APCReadPar(APC_FB_POS)
#18.12 = MULDIV(FeedbackPosCnts%, 10, 65536)

// Parameter Read Pointer
_P01%       = APCReadPar(_P00%)

GOTO TOP:

} //Background


Pos0{
// POS0 Task Runs Before APC Runs

// SET REFERENCE POSITION FROM CTSync
// *******************************************************************
// Get Virtual Master Reference from CTSync (Only using 32bit Position Counter)
(MasterReference1%, MasterReference2%, MasterControlWord%, Status%) =
CTSyncGetSlaveReferences()

// APC User Program Reference, Dissection of Virtual Counter
Turns%  = UnsignedTopWord(MasterReference1%)
Position% = UnsignedBottomWord(MasterReference1%)
PositionFine% = 0

// Set User Reference
UserRefStatus% = APCSetReferencePosition(Turns%, Position%, PositionFine%)

// MAIN PROGRAM
// *******************************************************************
// Enable APC with the Drive Reference ON
IF #01.11 = 1 THEN

    // Enable APC
    RunModeStatus% = APCSetRunMode(APC_ENABLE)

    IF MasterControlWord% = 1 THEN

        // Enable Digital Lock Reference
        PosSelStatus%= APCSelectReference(APC_DIG_LOCK_REF)

    ELSE
        // Set Position Reference (Stop)
        PosSelStatus%= APCSelectReference(APC_STOP_REF)

    ENDIF

ELSE

    // Disable APC - Track feedback position
    RunModeStatus% = APCSetRunMode(APC_DISABLE)

    // Set Position Reference (Stop)
    PosSelStatus%= APCSelectReference(APC_STOP_REF)

ENDIF

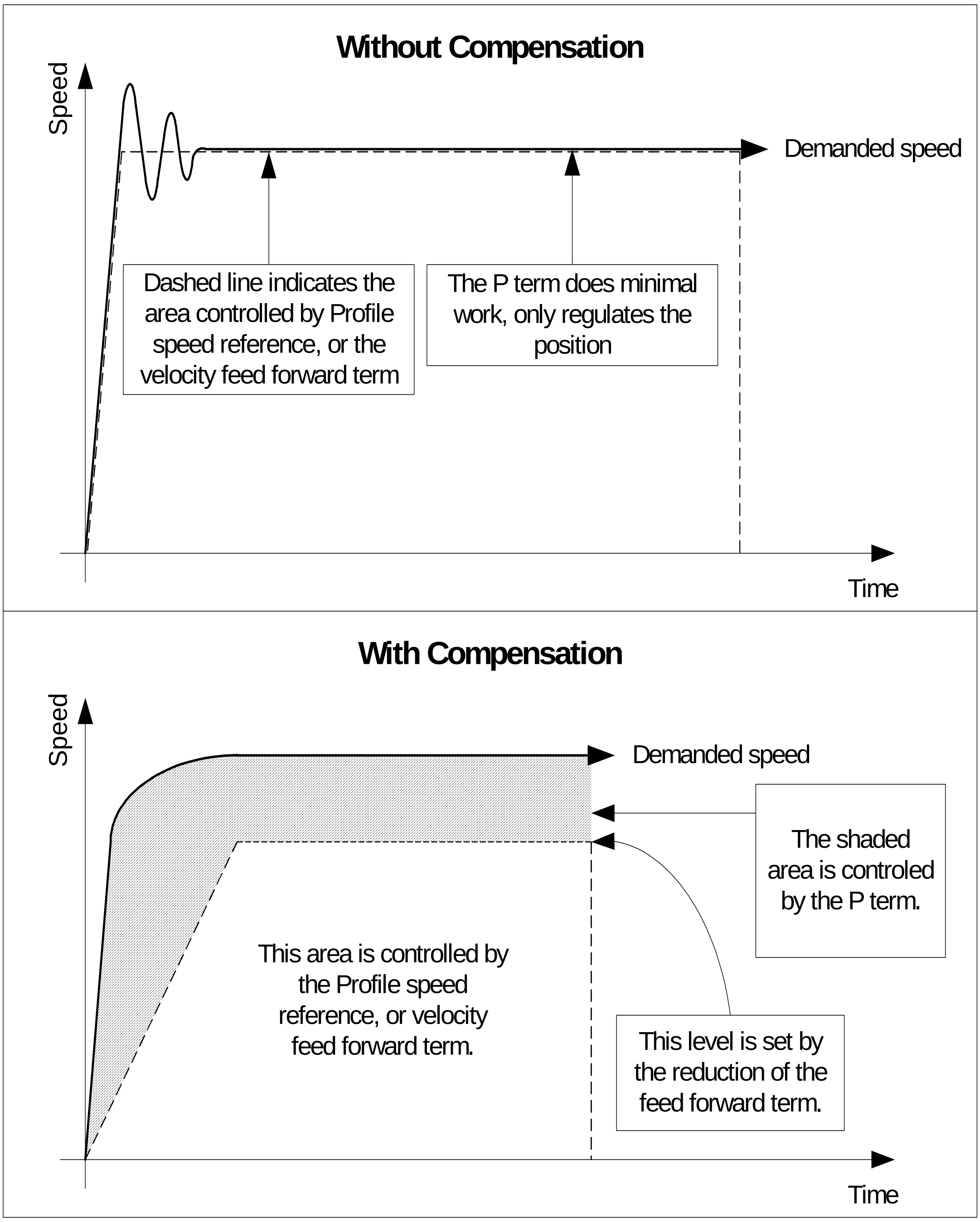

}POS0
```

8 Application Notes

8.1 Compensation For Overshoot With High Inertia Load

In applications where a high inertia is present, it may be desirable to dampen the response of the APC in order to reduce overshoot. Normally this can be achieved by introducing S ramps, to smooth out speed performance as the target speed is reached, however they are not available in this version of the APC. The method of dampening overshoot in this application note is implemented by reducing the speed feed forward term, so that the P term regulates a larger proportion of the demanded speed and position, and therefore the load is accelerated to the demand speed "softly". See Figure 8-1 below:

Figure 8-1



8.1.1 Checking Speed Loop

Before using this method of dampening it is essential that the performance of the drives speed loop is checked. This should be done by monitoring analogue output 1 with an oscilloscope. The following parameters must be set in order to monitor the speed from analogue output 1:

- 7.19 = 3.02
- 7.21 = H.Spd / 3
- 10.38 = 100 or press reset (to activate changes to the analogue inputs)

Connect the oscilloscope to pins 9 and 11(0V).

NOTE

If analogue output 1 does not provide a sufficient output voltage for the demand speed, lower drive parameter 1.06 till it is approximately 10% above the demand speed. This will give more volts per rpm, as the scaling for the analogue output is based on the value of parameter 1.06.

To monitor the drives speed loop performance the P gain for the position loop within the APC must be set to zero by using the command APCSetPGain(Gain%) in the user code, where Gain% = 0. Removing the P gain means that APC no longer regulates the position of the motor, so the drive is regulating it's self.

Ensure the speed loop can control the load, by setting the fastest acceleration rate the application requires, and applying a step change in demand velocity, whilst monitoring the performance using the oscilloscope. If it can not control the load sufficiently, follow the instruction in the Optimisation section in the Unidrive SP user guide.

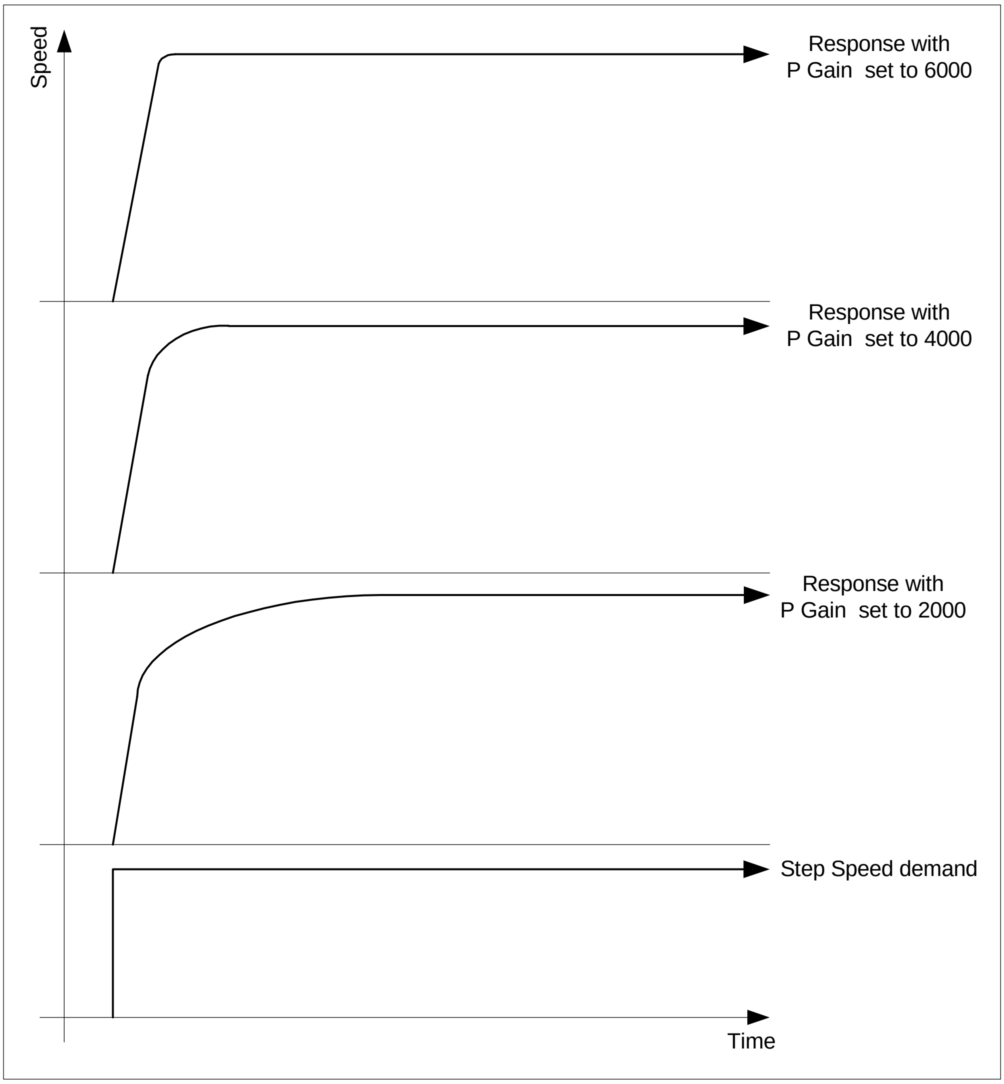
Once the Speed loop is tuned and the Position loop P gain is re-applied, it can be seen that an over shoot will occur which can not be tuned out by the P term. The compensation setup shown below resolves this problem.

8.1.2 Compensation Setup

1. The speed feed forward gain must be altered to less than 1. The Default settings for the speed feed forward is 1000, which gives a gain of 1 (1000/1000); setting the numerator to a value of 960, will provide good results for most applications. The command APCSetSpeedFFwdGain(Gain%) is used to set the speed feed forward gain value, where Gain% is a value of 960.

2. Using the oscilloscope to monitor performance, apply the same step change in speed demand as previously used, and set the value of the P Gain to get the desired damping level. Figure 8-2 shows an interpretation of the likely dampening effect with different P Gain settings:

Figure 8-2



.

NOTE This application note is based on laboratory tests using a 75UMD300CACAA Unimotor, with 4.8:1 load inertia to motor inertia ratio.

The speed loop gains were manually optimised for best performance, with an APC profile acceleration / deceleration rate of 80000 (equivalent to 4.47rpm /250µs). The gains shown in Figure 8-2 are actual values used in the laboratory test.

8.2 Position Loop Control on Open Loop Unidrive SP

In retro fit situations where an open loop application like a conveyor needs position control, and it is not possible to fit an encoder to the motor, but one can be installed on the line, the APC may be used close the position loop.

The example code in Figure below demonstrates how to use the APC in this way, and uses the APC speed reference to create the demand, (This can easily be replaced with another reference).

```
Notes{

This program demonstrates the use of the APC with Unidrive SP in Open Loop Vector
control, to provide crude position control for retro fit type applications, where an
encoder can not be fitted to the application motor.

Speed Reference
==================
The speed setpoint is set in 0.1rpm units entered into parameter 18.11.

Motion Profile Set up
=====================
The profile speed is 90% of the synchronous speed set for the application motor in
SynchronousSpeed%.
In this example it is set to 3000rpm, where the profile speed will be set to 2700rpm,
as the code has been written assuming a 2 pole motor is used.
The acceleration and deceleration is set to 0.5sec to get to or from 2700rpm and
0rpm.

Encoder Resolution
==================
Any encoder feedback can be used with this program, the number of counts per
revolution is 65536.

Drive Enable interlock
======================
When the APC run mode is interlocked with the drive Reference 0n parameter #01.11.
This parameter is set to 1 when the drive is healthy, Enabled and a run is commanded.

Drive & Motor Setup
====================
Before this program can be used the following must be configured: -
1. Drive run and enable control.
2. Enter motor map data & perform an autotune.
3. SM-Applications must be in a run state (x.13 = 1)

OTHER NOTES
===========
If a motor with more than 2 poles is used, the conversion factor SyncronousSpeed%
must be changed
to reflect the application motors Synchronous Speed. See the list below:
2 Pole = 3000
4 Pole = 1500
6 Pole = 1000
8 pole = 750

Setting this variable also defines the APC profile maximum speed clamp, and the P
Gain speed clamp.

This code uses one of the 3 CTSync output channels to send the trim speed to the
drive, so that the Reference
and Trim are written to the drive using the output channel method, and will be
actioned at the same time.

The Proportional gain set by parameter 18.12 must be set to use position control. It
is recommended that a start value of 1500 is used as high proportional gains tend to
lead to instability at high speeds.

To minimise following error seen in APC parameter [130], look at the following error
in a watch window, then raise and lower the motor rated frequency, #5.08, to make the
following error decrease to the optimum level.

} //Notes
```

```
Initial{

//Drive Setup Parameters
#01.06  = 50  //Max speed 50Hz
#01.07  = 0   //Min speed 0Hz
#01.10  = 1   //Bipolar reference selected
#01.14  = 5   //High precision reference selected
#02.11  = 0   //Set Accel ramp to zero
#02.21  = 0   //Set Decel ramp to zero
#02.04  = 0   //Fast ramps
#06.01  = 1   //Ramp stop selected

//SM-Applications setup Parameters
#81.06  = 25 //Enable CTSync comms mode (allows CTSync to write to drive parameters)
#81.12  = 3  //Set POS task update time to 1ms
#81.14  = 1  //Enable global runtime trip
#81.16  = 1  //Set encoder source update to every POS task
#81.17  = 0  //Disable parameter overrange trips
#81.20  = 1  //Enable power down save
#81.38  = 1  //Disable APC run time error trip
REINIT

//Configure then enable the APC and CTSync Output Channel (writing to precision
reference and trim, parameters 1.18 and 1.19)
APCChStatus1% = APCSetupOutputChannel(4,1,18)//APC output channel writes to
Reference
APCChStatus2% = APCEnableOutputChannel()
SyncChStatus1%= CTSYNCSetupOutputChannel(1,1,19)//CTSync output channel writes to
Trim
SyncChStatus2%= CTSYNCEnableOutputChannel(1)

//Configure the APC Feedback and Reference Source encoders
SourceStatus1% = APCSetFeedbackSource(APC_DRIVE_ENC) // Drive encoder selected
SourceStatus2% = APCSetReferenceSource(APC_USER_ENC) // User Program Selected
RunModeStatus% = APCSetRunMode(APC_DISABLE) // Primes the position counters on
startup or reset

//Set APC output ratio to compensate for output gear box ratio. Where In:Out =
Numerator:Denominator
RatioStatus1% = APCSetOutputRatioNumerator(1000)
RatioStatus2% = APCSetOutputRatioDenominator(1000)

//Synchronous speed for conversion from rpm to Hz, Calculated from 120*Motor Hz/Motor
Poles
SynchronousSpeed% = 3000

// Profile Max Speed (90% of motor application motor synchronous speed)
ProfMaxSpeed% = MULDIV(1073741824, (SynchronousSpeed% * 90), (60000 * 100))
ProfMxSpStatus%= APCSetProfileMaxSpeedClamp(ProfMaxSpeed%)

// P Gain Speed Clamp(10% of motor application motor synchronous speed)
PGainClamp% = MULDIV(1073741824, (SynchronousSpeed% * 10), (60000 * 100))
ProfMxSpStatus%= APCSetPGainSpeedClamp(PGainClamp%)

} //Initial
```

```
Background{
top:

// Speed Setpoint Scaling from 0.1rpm entered in parameter 18.11 to Internal units
SpeedSetpointInternal% = MULDIV(1073741824,(#18.11/10),60000)

//Send Speed Reference to APC
SpeedRefStatus% = APCSetSpeedSetPoint(SpeedSetpointInternal%)

//Set Position loop Proportional gain (try at 1500 to start)
APCSetPGain(#18.12)

goto top: // main background loop
} //Background

Pos0{

//Drive Reference on interlock
IF #01.11 = 1 THEN
    RunModeStatus% = APCSetRunMode(APC_ENABLE)//APC fully functional
ELSE
    RunModeStatus% = APCSetRunMode(APC_DISABLE)//Only APC source counters active
ENDIF

//18.31 selects either the Stop or Speed Reference
IF #18.31 = 1 THEN
    RefSelectStatus% = APCSelectReference(APC_SPEED_REF)
ELSE
    RefSelectStatus% = APCSelectReference(APC_STOP_REF)
ENDIF

} //Pos0

Pos1{

//Get high resolution speed reference from the APC position loop output
Speed% = APCGetOutputSpeed()

//Convert high resolution reference to rpm x1000 units to maintain resolution in
integer maths
Rpmx1000% = MULDIV(Speed%,60000000,1073741824)

//Convert rpm x1000 into Hz x1000 (open loop drive uses Hz not rpm)
Hzx1000% = MULDIV(Rpmx1000%,50,SynchronousSpeed%)

//Remove trim from HZx1000% to give Reference only
ReferenceHz% = MULDIVRM(Hzx1000%,1,100)

//Remove Reference from HZx1000% to give trim only
TrimHz% = Hzx1000%-(ReferenceHz%*100)

//Modify drive reference to account for #1.19 being a unipolar value
IF TrimHz% < 0 THEN

    //Find the difference between full scale and TrimHz%
    TrimHz% = 100 + TrimHz%

    //Subtract 1 from ReferenceHz so that the overall reference is the same value
    ReferenceHz% = ReferenceHz% - 1

ENDIF

//Send calculated Reference and Trim to drive
APCWriteOutputChannel(ReferenceHz%)
CTSYNCWriteOutputChannel(1,TrimHz%)

//General Read parameter
Value% = APCReadPar(Read%)

} //Pos1
```

8.3 Conversion and Word Manipulation

With the introduction of the APC comes a number of useful Conversion Functions:

8.3.1 Embedded APC Converter

To assist the user to perform Position Control operations in familiar units, conversion from user to APC units, and from APC to user units is provided. The user has to specify the number of units required for a single revolution of the motor (UPR), after which the position or distance, velocity, and acceleration can be specified in user units, user units/s, and user units/s/s respectively.

The example code below demonstrates how the embedded APC conversion blocks may be applied:

```
Notes{
This example code demonstrates how the APC embedded velocity conversion block may be
used to convert a speed reference in counts/s to an APC speed reference, and how the
APC position loop speed output may be converted back into counts/s for monitoring.
} //Notes

Initial{
//Initialise conversion variable
CounsPerSecIn%

} //Initial

Background{
top:

// Set the converter UPR (Units Per Rev). For conversion to counts/s, the resolution
// set by the number of turns bits should be used. for this example 16bit resolution
// is assumed.
SetUPR(65536)

// Convert user reference, UserVelocity%, in counts/s into an APC velocity,
// APCVelocity%
(APCVelocity%,ConvertStatus1%)     = UserToAPCVelocity(CounsPerSecIn%)

//Send converted velocity to APC speed reference
SpeedStatus%                      = APCSetSpeedSetpoint(APCVelocity%)

//Get position loop output and covert to counts/s
(APCSpeedOut%,ReadStatus%)        = APCReadPar(135)
(CountsPerSecOut%,ConvertStatus2%) = APCToUserVelocity(APCSpeedOut%)

goto top: // main background loop
} //Background
```

See section 5.3.1 *Embedded APC Converter* for more details.

8.3.2 User Defined Unit Converter

This converter is not intended exclusively for use with the APC, but can be used for any purpose. For example millimeters to inches conversion and back again can be performed by setting the numerator to 254 and the denominator to 10. Converting forwards will scale by 25.4, which will convert inches to millimeters. Converting backwards will convert millimeters to inches. The following example code demonstrates this:

```
Notes{
This example code demonstrates how the user defined conversion block may be used to
convert a value in inches to millimeters and vice versa.
} //Notes

Initial{

//Initialise conversion variables
ConvertInches%  = 0
ConvertMm%      = 0

} //Initial

Background{

top:

//Set converter numerator and denominator
SetConverterNumerator(254)
SetConverterDenominator(10)

//Convert inches to mm
AnswerInMm%     = ConvertForwards(ConvertInches%)

//Convert mm to inches
AnswerInInches% = ConvertBackwards(ConvertMm%)

goto top: // main background loop

} //Background
```

See section 5.3.2 *User Defined Unit Converter* for more details.

8.3.3 Word Manipulation Function Blocks

V01.03.00 introduces several useful 32bit word manipulation function blocks:

UnsignedTopWord - This command takes a 32-bit integer value, and converts the upper 16-bits to an unsigned 16-bit value, in the lower 16-bits of the output variable.

SignedTopWord - This command takes a 32-bit integer value, and converts the upper 16-bits to a Signed 16-bit value, in the lower 16-bits of the output variable.

UnsignedBottomWord - This command takes a 32-bit integer value and converts the lower 16-bits to an Unsigned 16-bit value, in the lower 16-bits of the output variable.

SignedBottomWord - This command takes a 32-bit integer value, and converts the lower 16-bits to a signed 16-bit value, in the lower 16-bits of the output variable.

PutTopWord - This command takes the lower 16 bits of the 32-bit integer value from the first input variable, and puts them in to the upper 16bits of the output variable. It also takes the lower 16 bits of the 32-bit integer value from the second input variable, and puts them in to the lower 16bits of the output variable.

PutBottomWord - This command takes the lower 16 bits of the 32-bit integer value from the first input variable, and puts them in to the lower 16bits of the output variable. It also takes the upper 16 bits of the 32-bit integer value from the second input variable, and puts them in to the upper 16bits of the output variable.

One application of these function blocks is using them to dissect a 32bit position sent via CTSync, into its component parts ready for use by the APC, which requires separate Turns, Coarse, and Fine position data when entered via the User Program reference. The example code below shows how this may be achieved:

```
Notes{
This example code demonstrates how dissect a 32bit position word consisting of Turns
(Upper 16bits), and Coarse Position (Lower 16bits) transmitted via CTSync, and send
the dissected information to the Reference source user program reference.
} //Notes

POS0{

// Get Reference from CTSync (Only using 32bit Position Counter)
(MasterReference%, Spare%, MasterControlWord%, Status%) = CTSyncGetSlaveReferences()

// APC User Program Reference, from Dissection of MasterReference1%
Turns%         = UnsignedTopWord(MasterReference%)
Position%      = UnsignedBottomWord(MasterReference%)
PositionFine%  = 0

// Send dissected position information to the Reference source, user program
reference
UserRefStatus% = APCSetReferencePosition(Turns%, Position%, PositionFine%)

} //POS0
```

See section 5.3.3 *Word Manipulation Function Blocks* for more details.

8.4 Remainder controlled Virtual Master

For many applications a basic Virtual Master is adequate, however in applications where the Virtual Master speed must be precise, it is possible to look after any remainder from the speed conversion calculation, and maintain an accurate and steady speed. The example below shows what happens without remainder control:

Pos task Cycle	= 1ms
Demand speed in RPM	= 100
CPR	= 65536
Counts Per Pos Task Cycle	= RPM * CPR / 60000
	= 100 * 65536 / 60000
	= 109.2266667

In DPL This looks like `MULDIV(SpeedRPM%,CPR%,60000)`.

Only the whole counts per Pos task sample can be represented, therefore the speed will be lower than expected, when the Virtual Master Counter value is incremented by this value every Pos task cycle.

To solve this problem, the `MULDIVRM` DPL command may be used to give whole counts per Pos task sample and a remainder. If the remainder is accumulated every Pos task cycle, as soon as a whole counts worth of remainder has been accumulated e.g. 60000 for the example above, an extra count can be added to the Virtual Master counter value, and the appropriate value subtracted from the remainder accumulator. The coded example below shows how to implement this type of Virtual Master:

```
Notes{
This example code demonstrates how to create a remainder controlled Virtual Master
} //Notes

Initial{

RPM%              = 100
CountsPerRev%     = 65536
VMRemainderCount% = 0
VMCounter%        = 0

} //Initial

Pos0{

// Virtual Master Counter
// Calculate counts per Pos task for virtual master generator
// (Demand speed in RPM * CPR) / (60 * Pos task samples per second)
(VMCntsPerSample%,VMRemainder%) = MULDIVRM(RPM%,CountsPerRev%,60000)

//Virtual Master Remainder accumulator
VMRemainderCount% = VMRemainderCount% + VMRemainder%

//VM Remainder position compensation
//If Virtual Master count is >= than (60 * Pos task samples per second)
IF VMRemainderCount% >= 60000 THEN

    //Subtract denominator from count
    VMRemainderCount% = VMRemainderCount% - 60000

    //Add 1 count to VMCntsPerSample%
    VMCntsPerSample% = VMCntsPerSample% + 1

//If Virtual Master count is <= than -(60 * Pos task samples per second)
ELSEIF VMRemainderCount% <= -60000 THEN

    //Subtract denominator from count
    VMRemainderCount% = VMRemainderCount% + 60000
```

```
    //Add 1 count to VMCntsPerSample%
    VMCntsPerSample% = VMCntsPerSample% - 1

ENDIF

//Virtual Master Position accumulator
VMCounter% = VMCounter% + VMCntsPerSample%

// Split up Virtual Master Counter data ready for the APC User Program Reference
Turns% = UnsignedTopWord(VMCounter%)
Position% = UnsignedBottomWord(VMCounter%)
PositionFine% = 0

// Send Virtual Master counter data to the APC User Program Reference
APCSetReferencePosition(Turns%, Position%, PositionFine%)

} //Pos0
```

9 Migration and Software changes

9.1 Migration from V01.02.01 Firmware

Users familiar with SM-Applications firmware Version V01.02.01 need to be aware of fundamental changes and addition to the virtual parameters with the introduction of the APC in firmware version >= V01.03.00.

9.1.1 Virtual Parameters that have changed function

Table 9-1

Parameter	<= V01.02.01 Description	>= V01.03.00 Description
81.16	Auxiiary encoder select	Encoder data update rate
81.38	Reserved	Disable APC runtime error
90.01	Main encoder position	Feedback encoder position
90.02	Main encoder revolution count	Feedback encoder revolution count
90.03	Auxiliary encoder position	Reference encoder position
90.04	Auxiliary encoder revolution count	Reference encoder revolution count
90.18	Enable freeze input	Feedback encoder freeze flag
90.19	Freeze encoder position count	Feedback encoder freeze position
90.20	Freeze encoder revolution count	Feedback encoder freeze revolution count
90.25	Main encoder marker position	Feedback encoder marker position
90.26	Main encoder marker revolution count	Feedback encoder marker revolution count
90.28	Auxiliary encoder freeze flag	Reference encoder freeze flag
90.29	Auxiliary encoder freeze position	Reference encoder freeze position
90.30	Auxiliary encoder freeze revolution count	Reference encoder freeze revolution count

9.1.2 New Virtual Parameters in >= V01.03.00 Firmware

Table 9-2

Parameter	Description
90.31	Feedback encoder turns and coarse position
90.32	Reference encoder turns and coarse position
90.33	Feedback encoder freeze turns and coarse position
90.34	Reference encoder freeze turns and coarse position
90.35	Reference encoder marker position
90.36	Reference encoder marker revolution count
90.37	Feedback encoder marker turns and coarse position
90.38	Reference encoder marker turns and coarse position
90.41	Reference encoder marker flag
90.42	Feedback encoder marker flag
90.43	Reference encoder source

Parameter	Description
90.44	Feedback encoder source
90.45	Reference marker flag enable
90.46	Feedback marker flag enable
90.47	Reference freeze enable
90.48	Reference freeze enable
91.17	Number of valid CTSync messages
91.18	Number of invalid CTSync messages
91.19	Number of missing CTSync messages
91.20	Synchronisation signal too short
91.21	Inter option module synchronisation control
91.22	Inter option module synchronisation status

NOTE Most new users testing firmware version V01.03.00 find that after downloading, they are unable to get position feedback in 90.01 and 90.02. This is because in this firmware, the drive encoder is not automatically directed to 90.01 and 90.02 two new parameters have been created to direct both the Reference (Auxiliary) and Feedback (Main) encoder sources:

90.43 - Which effectively replaces 81.16, and sets the Reference encoder source. The reference encoder position data is displayed in 90.03 and 90.04.

90.44 - Which sets the Feedback encoder source. The Feedback encoder position data is displayed in 90.01 and 90.02.

For both parameters 90.43 and 90.44, a value from 0 to 5 can be set, where:

0 = Drive encoder

1 = Slot 1

2 = Slot 2

3 = Slot 3

4 = User program source (CTSync etc.)

5 = None

Therefore to view the drive encoder position data in 90.01 and 90.02, 90.44 must be set to 0. In Firmware version V01.03.00, 90.43 and 90.44 are set to a default value of 5, which means that from default no encoder data will be displayed in 90.01 to 90.04. In firmware version >V01.03.00, the default has been set to 0, therefore the drive encoder data will be displayed in 90.01 to 90.04.

9.2 APC Software Changes

9.2.1 APC Changes introduced with V01.03.03 Firmware

In V01.03.03 the following additional features have been added:

1. The Offsets have been moved downstream from the Digital Lock reference, so that all references except the Stop reference may use the offsets. This includes the addition of a second “Offset” Profile generator so that offsets can be applied smoothly.
2. A Speed Feed Forward term has been added

3. The Output Speed Clamp has been separated from the Profile Maximum Speed Clamp, and the P Gain Speed Clamp, so that it can be adjusted individually.
4. The ability to insert function blocks like filters, between the Source and Main counters has been added.
5. The encoder sources are now updated a second time at the end of POS0. This means that changes to the position information actioned in POS0, will be used by the APC in the same POS task cycle.

NOTE Because the position data sources are now updated twice in the same POS task cycle, it is important to set #81.16 to 1, update position sources every POS task. Failure to do this could result in position data skew.

6. DPL calls have been added to enable or disable the marker and freeze functions. These calls are the equivalent of setting/resetting #90.45 to #90.48.

9.2.2 APC Changes introduced with V01.03.04 Firmware

1. Read parameters [142] and [143] have been added. [142] shows the total profile position reference (117 + 157). [143] shows the total profile speed reference (118 +158).
2. Read parameter 4 has been added to show when the APC has been enabled. This parameter differs from read parameter 0 in that read parameter [4] is set/reset at the end of the APC run time, rather than before like read parameter 0. This is useful when conditional code is used to detect when the APC is enabled in the Pos0 task.
3. Read parameter [5] has been added to show the reset status. When set to 1 a reset is in progress, and when set to 0 again the reset is completed. This is useful when conditional code is used to detect when the APC is in reset in the Pos0 task.

9.2.3 APC Changes introduced with V01.04.01 Firmware

1. Read parameter [87] added to show when a single shot CAM has been completed. When set to 1 the CAM has completed.
2. Read parameter [88] and control commands APCEnableCAMFreezeRearm() and APCDisableCAMFreezeRearm(). This feature allows the user to make simple registration systems, by resetting the action on freeze flag, the CAM reference, and the freeze flag, such that after a reference freeze starts a single shot CAM, when the CAM has ended the system is reset and will wait for then next reference freeze. when read parameter [88] = 1 the feature is enabled
3. Read parameter 89 has been introduced to let the user know that the CAM has wrapped/rolled over in this position task sample. This is set at the end of the APC run time, and can be read in Pos 1 first.
4. All ratio numerators and denominators, including CAM output ratio, Digital lock ratio and APC output ratio are now 31bit
5. A Reference reset position offset has been added to allow the user to offset the reference main APC position counter [28]. The reference offset [6] is set using APCSetRefPosResetOffset(Position%).
6. Four additional output channel modes have been added, which allows the user to invert the speed reference sent to the drive:

   10 = Hard Speed Reference Inverted

   12 = Speed Reference (#1.21) Inverted

   13 = User Parameter Selection Inverted

   14 = User Reference Inverted

These are useful if the user needs to invert the Feedback encoder data. Using the inverted mode corrects the position loop output direction to match the feedback inversion.

9.2.4 APC Changes introduced with V01.04.04 Firmware

The APC has some new features added to enable the APC to operate directly in user units. The new features are available by using SM-Applications firmware =>V01.04.04, and Sypt Pro =>V02.00.11.This new functionality allows the user to:

- Enter and read position in user units, with remainder to prevent loss of position information
- Enter and read speed in user units per second
- Enter and read acceleration / deceleration in user units per second per second
- No need to convert externally using the APCToUser and UserToAPC conversion method (saves an additional command call)
- All positioning references may be controlled in user units (except the CAM reference which will be implemented soon.

User units are whatever unit of position measurement a system is running in e.g. mm, inches, bottles, sheets etc. If for example mm are the chosen position unit, then speeds will be in mm/s, and acceleration will be in mm/s/s.

10 Glossary of Terminology

Absolute Move

A move referenced from a fixed absolute zero position.

Acceleration

This is the rate of change of velocity. Acceleration has two characteristics: magnitude and direction. Name commonly used as positive acceleration, i.e. going from a lower velocity to a higher velocity

Accuracy

This is an absolute measurement defining the difference between expected and actual position.

Actual Position

This is the position of an axis relative to the commanded position. This may be the position at the end of the commanded move or the lag between command position at any point during the move and the actual position of the axis at that point. The later is commonly referred to as following error.

Axis

A principal direction along which movement of a tool, component or work piece occurs.

Backlash

This is the amount of play, (lost motion), between a set of moveable parts when changing the direction of travel. Typically seen in drive trains, lead screws, & bearings.

Centralised Control

This is a control system in which all of the primary processing is done at a single location rather than at multiple points throughout the system.

Closed Loop

This is a positioning system, which employs feedback information to regulate the output response.

Co-ordination

The integration of the movements of two or more axes of motion so that the resultant motion is a path which none of the axes are capable of independently. Coordination may also involve the use of sensors and other internal or external commands in the Integration effort, which assist in effecting the movement or work desired.

Deceleration

This is the rate of change of velocity. Name commonly used as negative acceleration, i.e. going from a higher velocity to a lower velocity

Decentralised control

A control system in which the logic functions and input/output functions are located at individual pieces of equipment or sub systems and function essentially independent of each other. Normally the independent systems will have some means of communicating vital information with each other.

Dwell Time

This defines the time within a move cycle where no motion occurs.

Electronic CAM profiles

A technique used to perform non-linear motion electronically similar to that achieved with mechanical cams.

Electronic gearing

This is a method that simulates mechanical gears by electrically synchronising one closed loop axis to a second axis (open- or closed-loop) through a variable ratio.

Encoder

An electromechanical device, which produces discrete electrical pulses directly, related to the angular position of the input shaft, providing high-resolution feedback data on position, velocity, and direction.

Encoder Resolution

The number of electrically identified positions occurring in 360 degrees of input shaft rotation.

Feedback

The signal or signals received from a controlled machine or process to denote its response to the command signal.

Feed Forward

This is a method that "pre-compensates" a control loop for known errors due to motor, drive, or load characteristics to improve response. It depends only on the reference generated not the measured error.

Following Error

This is the difference between the reference position of an axis and its actual feedback position. The amount of Following Error present varies with the speed of the axis. The amount of following error allowed can be adjusted through the KV parameter.

Home Position

This is a reference position for all absolute positioning movements. Usually defined by a home limit switch and/or encoder marker. Normally set at power-up and retained as long as control system is operational.

Homing

Locating a unique reference position at power-up for axis calibration.

In/At Position Window

This is the range of position increments in which the axis is considered by the controller to be at the commanded position point. Can be thought of in terms of +/- N position increments from the commanded position.

Interpolation

This is a mathematical expression, which defines a coordinated move of an axis with respect to another.

Jerk limitation

Limits the rate of acceleration change during the movement of an axis. Its purpose is to eliminate mechanical jerking when speed changes are made.

Jog

This is an axis running at a fixed velocity and acceleration/deceleration rate, in a selected direction, with no specific destination.

KP

Velocity Loop Proportional Gain; determines how much velocity error will be allowed by the servo system during a move. See also: Tuning

KV

Position Loop Gain; determines how much positioning error, or following error, will be allowed by the servo system during a move. See also: Tuning

Linear

This is the relationship between an input and output in which the output varies in direct proportion to the input.

Loop Update Time

This defines the time interval between updates to calculate the process set point from the following error.

Master

This is a physical feedback, which provides position information for a synchronized axis to follow.

Motion Profile

This is a method of describing a move operation in terms of time, position, and velocity. Typically velocity is characterised as a function of time or distance, which results in a triangular or trapezoidal profile.

Offset

A preset distance between the actual zero reference point and a programmed zero reference point.

Open Loop

This is a system, which does not employ feedback information.

Overshoot

This is a system response where the output or result exceeds the desired value.

Phasing

Adjusting the position of one axis with respect to others during synchronisation or electronic gearing. This is usually done while the axes are moving, and done to correct for small registration problems.

Position Error

Error caused when the difference between the actual position, and the command position is greater than a set amount.

Positioning

Specifying a move by giving a target position, a velocity and acceleration. The target position can be an absolute position, or a relative position from the current position.

Position Loop

Portion of the command signals that generates the position information based on position feedback.

Programming Language

This is the interface that allows the user to control the motion system according to the demands of the user.

Quadrature

This is a technique that separates signal channels by 90' (electrical) in feedback devices. It is used with encoders and resolvers to detect direction of motion.

Relative Move

A move referenced from the current set position.

Repeatability

This is the ability of a positioning system to return to an exact location during operation (from the same direction with the same load and speed).

Resolution

The smallest positioning increment achievable. In digitally programmed systems it is the smallest specifiable positioning increment.

S curve

S curve refers to a control pattern that accelerates and decelerates a motor slowly to reduce mechanical shock. This function is more sophisticated than linear acceleration.

Synchronisation

The condition that occurs when several functions of a machine (mechanical, servo or software) follow a common control signal and are in a specific position according to this signal.

Task

A software system control that determines the execution rates and priority levels for software modules running in a motion control or PLC.

Tuning

Adjusting the servo drive's internal characteristics to give it the ability to control the reflected inertia and gives the axis a smooth position/velocity profile. The process of Tuning involves setting the Velocity Loop Proportional Gain (KP), Position Loop Gain (KV), and the Velocity Loop Integral Action (Ki) values so that the axis has a position/ velocity profile allowing only as much position/velocity error as the process will permit.

Velocity

This is the rate at which the position of a moving object is changing. Velocity has two characteristics: magnitude (speed) and direction.

Velocity Loop

A servo control function that sums a velocity command signal with a speed feedback signal from a motor, and outputs the difference as a torque command signal.

Virtual Master

An encoder signal created in the software of a motion control to allow synchronising of multiple servo systems. A typical machine may have several virtual master encoders.

11 Quick Reference

11.1 APC Command Reference

11.1.1 Control and Access Functions

Command	Units	Range	Default
= APCSetRunMode (Mode%)	NA	0: Off 1: Disable 2: Enable	0
= APCReset ()	NA	Active or Inactive	Inactive
= APCGetOutputSpeed ()	2^{32} counts / rev encoder over 250µs / 250µs	+/- (Max profile speed + P speed clamp)	0
= APCGetOutputSpeedRpmx10 ()	rpm x 10	+/- (Max profile speed + P speed clamp)	0
= APCSelectAbsoluteMode ()	NA	Active or Inactive	Inactive
= APCSelectRelativeMode ()	NA	Active or Inactive	Active
= APCReadPar ()	Dependant on parameter	Dependant on parameter	Dependant on parameter
= APCReadParWithRem ()	Dependant on parameter	Dependant on parameter	Dependant on parameter
= APCSetPositionResetOffset (Position%)	Encoder counts or user units	-2^{31} to $2^{31}-1$	0
= APCSetRefPosResetOffset (Position%)	Encoder counts or user units	-2^{31} to $2^{31}-1$	0
= APCResetSourcesOnDisable ()	NA	Active or Inactive	Active
= APCDoNotResetSourcesOnDisable ()	NA	Active or Inactive	Inactive

11.1.2 Feedback and Reference Source

Command	Units	Range	Default
= APCSetReferenceSource (Source%)	NA	0: Drive 1: Slot 1 2: Slot 2 3: Slot 3 4: User program 5: Unconfigured	0

Command	Units	Range	Default
= APCSetFeedbackSource (Source%)	NA	0: Drive 1: Slot 1 2: Slot 2 3: Slot 3 4: User program 5: Unconfigured	0
= APCSetNumOfTurnsBits (TurnBits%)	NA	1 to 16	16
= APCEnableRefSourceMarker ()	NA	Active or Inactive	Inactive
= APCDisableRefSourceMarker ()	NA	Active or Inactive	Active
= APCEnableFbckSourceMarker ()	NA	Active or Inactive	Inactive
= APCDisableFbckSourceMarker ()	NA	Active or Inactive	Active
= APCResetRefSourceMarkerFlag ()	NA	Active or Inactive	Inactive
= APCResetFbckSourceMarkerFlag ()	NA	Active or Inactive	Inactive
= APCResetRefSourceFreezeFlag ()	NA	Active or Inactive	Inactive
= APCResetFbckSourceFreezeFlag ()	NA	Active or Inactive	Inactive
= APCEnableReferenceMarker ()	N/A	Active or Inactive	Inactive
= APCDisableReferenceMarker ()	N/A	Active or Inactive	Active
= APCEnableFeedbackMarker ()	N/A	Active or Inactive	Inactive
= APCDisableFeedbackMarker ()	N/A	Active or Inactive	Active
= APCEnableReferenceFreeze ()	N/A	Active or Inactive	Inactive
= APCDisableReferenceFreeze ()	N/A	Active or Inactive	Active
= APCEnableFeedbackFreeze ()	N/A	Active or Inactive	Inactive
= APCDisableFeedbackFreeze ()	N/A	Active or Inactive	Active
= APCInvertRefSource ()	NA	Active or Inactive	Inactive
= APCDoNotInvertRefSource ()	NA	Active or Inactive	Active
= APCInvertFbckSource ()	NA	Active or Inactive	Inactive
= APCDoNotInvertFbckSource ()	NA	Active or Inactive	Active
= APCSetReferencePosition (Turns%, Position%, PositionFine%)	Turns and Encoder counts	16bit Turns 16bit Position 16bit PositionFine	0
= APCSetFeedbackPosition (Turns%, Position%, PositionFine%)	Turns and Encoder counts	16bit Turns 16bit Position 16bit PositionFine	0

Command	Units	Range	Default
= APCGetReferenceStatus ()	NA	0: Drive 1: Slot 1 2: Slot 2 3: Slot 3 4: User Program 5: Unconfigured	0
= APCGetFeedbackStatus ()	NA	0: Drive 1: Slot 1 2: Slot 2 3: Slot 3 4: User Program 5: Unconfigured	0
= APCEnableRefInput ()	NA	Active or Inactive	Inactive
= APCDisableRefInput ()	NA	Active or Inactive	Active
= APCEnableFbckInput ()	NA	Active or Inactive	Inactive
= APCDisableFbckInput ()	NA	Active or Inactive	Active
= APCSetRefInput ()	Counts	32bit signed	Inactive
= APCSetFbckInput ()	Counts	32bit signed	Active

11.1.3 References

Command	Units	Range	Default
= APCSelectReference (Reference%)	NA	0: Stop 1: Position 2: Speed 3: CAM 4: Digital Lock	0
= APCSelectActionOnFreeze (Action%)	NA	0: No Action 1: CAM 2: Digital Lock	0
= APCSetStopMode (Mode%)	NA	0: Profiled Stop 1: Fast Stop	0
= APCTransferStopPosToPosSetPoint ()	NA	NA	NA
= APCSetPositionSetPoint (Position%)	Encoder counts or user units	-2^{31} to 2^{31}-1	0
= APCSetSetPointRem (Remainder%)	Signed positional remainder	-2^{31} to 2^{31}-1	0
= APCSetSpeedSetPoint (Speed%)	2^{32} counts / rev encoder over 250µs or user units/s	-715827883 to 715827883	0
= APCCamInitialise (InArray%, OutArray1%, OutArray2%, InterpolationArray%)	NA	NA	NA
= APCCamInitialise1 (InArray%, OutArray1%)	NA	NA	NA
= APCCamInitialise2 (InArray%, OutArray1%, InterpolationArray%)	NA	NA	NA
= APCCamInitialise3 (InArray%, OutArray1%, OutArray2%)	NA	NA	NA

Command	Units	Range	Default
= APCSetCAMStartIndex (StartIndex%)	NA	0 to 65535	0
= APCSetCAMSize (Size%)	NA	0 to 65535	0
= APCSetCAMDeltaSegLimit (DeltaLimit%)	Encoder counts / POS task clock time	0 to 65535	256
= APCSetCAMInterpolationMode (InterpolationMode%)	NA	-1: Multiple 0: Linear 1: Cosine	0
= APCSetCAMOutRatioNumerator (Ratio%)	NA	0 to 2147483647	1000
= APCSetCAMOutRatioDenominator (Ratio%)	NA	1 to 2147483647	1000
= APCSelectCAMAbsoluteReset ()	NA	Active or Inactive	Inactive
= APCSelectCAMZeroReset ()	NA	Active or Inactive	Active
= APCSetCAMAbsResetIndex (Index%)	NA	0 to 65535	0
= APCSetCAMAbsResetPositionInSeg (Position%)	Encoder counts	0 to 2^{31}-1	0
= APCEnableCAMSingleShot ()	NA	Active or Inactive	Inactive
= APCDisableCAMSingleShot ()	NA	Active or Inactive	Active

= APCSetDigLockMode (Mode%)	NA	0: Unlocked 1: Locked 2: Never locked	0
= APCSetDigLockRatioNumerator (Numerator%)	NA	0 to 2147483647	1000
= APCSetDigLockRatioDenominator (Denominator%)	NA	1 to 2147483647	1000
= APCSetDigLockLockingSpeed (Speed%)	2^{32} counts / rev encoder over 250µs or user units/s	0 to 2^{31}-1	17895
= APCSetDigLockLockingPosition (Position%)	Encoder counts or user units	0 to 2^{31}-1	6553
= APCEnableRigidLock ()	NA	Active or Inactive	Inactive
= APCDisableRigidLock ()	NA	Active or Inactive	Active

11.1.4 Profile Generators

11.1.4.1 Main Profile Generator

Command	Units	Range	Default
= APCEnableProfile ()	NA	Active or Inactive	Active
= APCDisableProfile ()	NA	Active or Inactive	Inactive
= APCSetProfileAccelRate (AccelRate%)	2^{32} counts / rev encoder over 250µs / 250µs or user units/s/s	0 to 4473924	22369
= APCSetProfileDecelRate (DecelRate%)	2^{32} counts / rev encoder over 250µs / 250µs or user units/s/s	0 to 4473924	22369
= APCSetProfileMaxSpeedClamp (MaxSpeed%)	2^{32} counts / rev encoder over 250µsor user units/s	0 to 715827883	48318382

11.1.4.2 Offset Profile Generator

Command	Units	Range	Default
= APCSetSpeedOffset (Speed%)	2^{32} counts / rev encoder over 250µs or user units/s	-715827883 to 715827883	0
= APCSetPositionOffset (Position%)	Encoder counts or user units	-2^{31} to $2^{31}-1$	0
= APCSetOffsetRem (Remainder%)	Signed positional remainder	-2^{31} to $2^{31}-1$	0
= APCSelectOffset (OffSelect%)	NA	0 to 2	0
= APCSetOffProfileAccelRate (AccelRate%)	2^{32} counts / rev encoder over 250µs / 250µs or user units/s/s	0 to 4473924	22369
= APCSetOffProfileDecelRate (DecelRate%)	2^{32} counts / rev encoder over 250µs / 250µs or user units/s/s	0 to 4473924	22369
= APCSetOffProfileMaxSpeedClamp (MaxSpeed%)	2^{32} counts / rev encoder over 250µs or user units/s	0 to 715827883	48318382

11.1.5 Position Loop

Command	Units	Range	Default
= APCSetPGain (Gain%)	0.01 radians / s / radian unit	0 to 65535	2500
= APCSetPGainSpeedClamp (PClamp%)	2^{32} counts / rev encoder over 250µs or user units/s	0 to 715827883	5368709
= APCEnableExternalRefSpeed ()	NA	Active or Inactive	Inactive
= APCDisableExternalRefSpeed ()	NA	Active or Inactive	Active
= APCSetExternalRefSpeed (Speed%)	2^{32} counts / rev encoder over 250µs or user units/s	-715827883 to 715827883	0
= APCEnableExternalRefPosition ()	NA	Active or Inactive	Inactive
= APCDisableExternalRefPosition ()	NA	Active or Inactive	Active
= APCSetExternalRefPosition (Position%)	Encoder Counts or user units	-2^{31} to $2^{31}-1$	0
= APCSetSpeedFFwdGain (Gain%)	NA	0 to 2000	1000
= APCSetOutputRatioNumerator (Numerator%)	NA	0 to 2147483647	1000

Command	Units	Range	Default
= APCSetOutputRatioDenominator (Denominator%)	NA	1 to 2147483647	1000
= APCSetOutputSpeedClamp (Clamp%)	2^{32} counts / rev encoder over 250µs or user units/s	0 to 715827883	53687091
= APCSetupOutputChannel (Mode%, Menu%, Parameter%)	NA	Mode%- 0: Hard speed 1: Reserved 2: Speed ref 3: Parameter 4: User ref 5: Quiet 10: Hard Speed inverted 12: Speed ref inverted 13: Parameter inverted 14: User ref inverted	5
= APCEnableOutputChannel ()	NA	Active or Inactive	Inactive
= APCDisableOutputChannel ()	NA	Active or Inactive	Active
= APCWriteOutputChannel (Value%)	NA	-2^{31} to 2^{31}-1	0

11.1.6 Word Manipulation

Command	Units	Range	Default
= UnsignedTopWord (Input%)	NA	-2^{31} to 2^{31}-1	NA
= SignedTopWord (Input%)	NA	-2^{31} to 2^{31}-1	NA
= UnsignedBottomWord (Input%)	NA	-2^{31} to 2^{31}-1	NA
= SignedBottomWord (Input%)	NA	-2^{31} to 2^{31}-1	NA
= PutTopWord (Input1%, Input2%)	NA	-2^{31} to 2^{31}-1	NA
= PutBottomWord (Input%, Input2%)	NA	-2^{31} to 2^{31}-1	NA

11.1.7 Conversion

Command	Units	Range	Default
= SetUPRNumeratorAndDenominator (N%,D%)	NA	N% = 0 to 2^{31}-1 D% = 1 to 2^{31}-1	NA
= GetUPRNumeratorAndDenominator ()	NA	NA	NA
= UserToAPCPosition (UserPos%)	NA	-2^{31} to 2^{31}-1	NA
= APCToUserPosition (Pos%)	NA	-2^{31} to 2^{31}-1	NA
= UserToAPCVelocity (UserVel%)	NA	-2^{31} to 2^{31}-1	NA
= APCToUserVelocity (Vel%)	NA	-2^{31} to 2^{31}-1	NA
= UserToAPCAcceleration (UserAccel%)	NA	-2^{31} to 2^{31}-1	NA
= APCToUserAcceleration (Accel%)	NA	-2^{31} to 2^{31}-1	NA
= SetConverterNumerator (Numerator%)	NA	-2^{31} to 2^{31}-1	NA
= SetConverterDenominator (Denominator%)	NA	-2^{31} to 2^{31}-1	NA

Command	Units	Range	Default
= ConvertForwards (Input%)	NA	-2^{31} to 2^{31}-1	NA
= ConvertBackwards (Input%)	NA	-2^{31} to 2^{31}-1	NA

11.2 APC Read Parameters

Parameter	Description	Units	Range	Default
0	Position controller enable	NA	0: APC Disable requested 1: APC Enable requested	0
1	Absolute position reset mode select	NA	0: Relative mode 1: Absolute mode	0
2	Feedback position offset applied during APC position reset	Encoder counts or User Units	-2^{31} to $2^{31}-1$	0
3	Reset the sources on position control disable	NA	0: Do not reset sources 1: Reset sources	1
4	APC Enable Status. Shows when the APC has Actually changed to enable.	N/A	0: APC Disabled 1: APC Enabled	0
5	APC Reset Status. Shows when a reset is completed on falling edge e.g. 1 to 0.	N/A	0: Reset Complete 1: Reset in progress	0
6	Reference position offset applied during APC position reset	Encoder Counts or User Units	-2^{31} to $2^{31}-1$	0
13	Pos task sample time	Milliseconds	0: 0.25 1: 0.5 2: 1 3: 2 4: 4 5: 8	
14	Alignment specified by number of turn bits	NA	1 to 16	16
15	Version number of position controller	NA	0 to 65535	NA
20	Reference source position	Encoder counts or User Units	-2^{31} to $2^{31}-1$	0
21	Reference source freeze position	Encoder counts or User Units	-2^{31} to $2^{31}-1$	0
22	Reference source freeze flag	NA	0: No reference freeze flag 1: Reference freeze flag	0
23	Reference source marker position	Encoder counts or User Units	-2^{31} to $2^{31}-1$	0
24	Reference source marker flag	NA	0 No reference marker flag 1 Reference marker flag	0
25	Reference source is affected by marker	NA	0: Reference is not affected 1: Referece is affected	0
26	Reference source speed	2^{32} counts / rev encoder over 250µs or User Units/s	-2^{31} to $2^{31}-1$	0
27	Reference source invert	NA	0: Do not invert reference 1: Invert reference	0
28	Integrated reference position	Encoder counts or User Units	-2^{31} to $2^{31}-1$	0
29	Integrated reference position on freeze	Encoder counts or User Units	-2^{31} to $2^{31}-1$	0
30	Integrated reference position on marker	Encoder counts or User Units	-2^{31} to $2^{31}-1$	0
31	Reference Input	Encoder counts only	-2^{31} to $2^{31}-1$	0

Parameter	Description	Units	Range	Default
32	Reference Input Disable	NA	0: Ref input not disabled 1: Ref input disabled	0
40	Feedback source position	Encoder counts or User Units	-2^{31} to $2^{31}-1$	0
41	Feedback source freeze position	Encoder counts or User Units	-2^{31} to $2^{31}-1$	0
42	Feedback source freeze flag	NA	0: No feedback freeze flag 1: Feedback freeze flag	0
43	Feedback source marker position	Encoder counts or User Units	-2^{31} to $2^{31}-1$	0
44	Feedback source marker flag	NA	0: No feedback marker flag 1: Feedback marker flag	0
45	Feedback source is affected by marker	NA	0: Feedback is not affected 1: Feedback is affected	0
46	Feedback source speed	2^{32} counts / rev encoder over 250µs or User Units/s	-2^{31} to $2^{31}-1$	0
47	Feedback source invert	NA	0: Do not invert feedback 1: Invert feedback	0
48	Integrated Feedback position	Encoder counts or User Units	-2^{31} to $2^{31}-1$	0
49	Integrated Feedback position on freeze	Encoder counts or User Units	-2^{31} to $2^{31}-1$	0
50	Integrated Feedback position on marker	Encoder counts or User Units	-2^{31} to $2^{31}-1$	0
51	Feedback Input	Encoder Counts only	-2^{31} to $2^{31}-1$	0
52	Feedback Input Disable	NA	0: Fbck input not disabled 1: Fbck input disabled	0
60	Rigid digital lock select	NA	0: Non-rigid lock 1: Rigid lock	0
61	Digital lock mode	NA	0: Unlocked 1: Locked 2: Never locked	0
62	Digital lock ratio numerator	NA	0 to 2147483647	1000
63	Digital lock ratio denominator	NA	1 to 2147483647	1000
64	Digital lock locking speed	2^{32} counts / rev encoder over 250µs or User Units/s	0 to $2^{31}-1$	17895
65	Digital lock locking position	Encoder counts or User Units	0 to $2^{31}-1$	6553
74	Start index of active CAM	NA	0 to 65535	0
75	Number of index values used by active CAM	NA	0 to 65535	0
76	Maximum change of segments allowed per sample	NA	0 to 65535	256
77	CAM interpolation type	NA	-1: Multiple interpolation 0: Linear interpolation 1: Cosine interpolation	0
78	CAM single shot mode select	NA	0: Cyclic mode 1: Single-shot mode	0

Parameter	Description	Units	Range	Default
79	CAM absolute reset mode	NA	0: CAM zero reset 1: CAM absolute reset	0
80	CAM output ratio numerator	NA	0 to 2147483647	1000
81	CAM output ratio denominator	NA	1 to 2147483647	1000
82	Current segment of active CAM	NA	0 to 65535	0
83	Delta input position from start of current segment	Encoder counts only	0 to 2^{31}-1	0
84	Size of CAM arrays	NA	0 to 65535	0
85	CAM absolute mode index reset value	NA	0 to 65535	0
86	CAM absolute mode position in segment reset value	Encoder counts only	0 to 2^{31}-1	0
87	The CAM has completed a single shot cycle.	N/A	0: Single shot not complete 1: Single shot complete	0
88	Indicates whether single shot freeze re-arm is selected or not.	N/A	0: Feature Disabled 1: Feature Enabled	0
89	Indicates that the CAM has wrapped / roller over this cycle	N/A	0: Not wrapped over 1: Wrapped over	0
90	Stopping mode	NA	0: Profiled stop 1: Fast Stop	0
91	Position the axis will stop at when the stop reference is selected	Encoder counts or User Units	-2^{31} to 2^{31}-1	0
92	Position set point	Encoder counts or User Units	-2^{31} to 2^{31}-1	0
93	Position offset	Encoder counts or User Units	-2^{31} to 2^{31}-1	0
94	Speed set point	2^{32} counts / rev encoder over 250µs or User Units/s	-715827883 to 715827883	0
95	Speed offset	2^{32} counts / rev encoder over 250µs or User Units/s	-715827883 to 715827883	0
96	APC internal converter numerator	N/A	0 to 2147483647	1
97	APC internal converter denominator	N/A	1 to 2147483647	1
100	Currently selected reference	NA	0: Stop 1: Position 2: Speed 3: CAM 4: Digital Lock	0
101	Action to reference selector on freeze	NA	0: No action 1: Digital lock on freeze 2: CAM on freeze	0
102	Offset Selector	NA	0: Offsets Disabled 1: Position Offset Enabled 2: Speed Offset Enabled	0
110	Profile acceleration rate	2^{32} counts / rev encoder over 250µs / 250µs or User Units/s/s	0 to 4473924	22369

Parameter	Description	Units	Range	Default
111	Profile deceleration rate	2^{32} counts / rev encoder over 250µs / 250µs or User Units/s/s	0 to 4473924	22369
113	Profile maximum speed	2^{32} counts / rev encoder over 250µs or User Units/s	0 to 715827883	48318382
114	Profile disable	NA	0: Enable profile 1: Disable profile	0
115	Profile input position	Encoder counts or User Units	-2^{31} to 2^{31}-1	0
116	Profile input speed	2^{32} counts / rev encoder over 250µs or User Units/s	-2^{31} to 2^{31}-1	0
117	Profile output position	Encoder counts or User Units	-2^{31} to 2^{31}-1	0
118	Profile output speed	2^{32} counts / rev encoder over 250µs or User Units/s	-2^{31} to 2^{31}-1	0
119	Profile output acceleration	2^{32} counts / rev encoder over 250µs / 250µs or User Units/s/s	-2^{31} to 2^{31}-1	0
120	At Target Position	NA	0: Not at target position 1: At target position	0
121	At Target Speed	NA	0: Not at target speed 1: At target speed	0
130	Position Error	Encoder counts or User Units	-2^{31} to 2^{31}-1	0
131	Position controller proportional gain	0.01 radians / s / radian unit	0 to 65535	2500
132	Position controller proportional gain maximum speed	2^{32} counts / rev encoder over 250µs or User Units/s	0 to 715827883	5368709
133	Position controller output ratio numerator	NA	0 to 2147483647	1000
134	Position controller output ratio denominator	NA	1 to 2147483647	1000
135	Position controller output speed	2^{32} counts / rev encoder over 250µs or User Units/s	-2^{31} to 2^{31}-1	0
136	External position reference select	NA	0: Profile generator output 1: External position ref	0
137	External reference position	Encoder counts or User Units	-2^{31} to 2^{31}-1	0
138	External speed reference select	NA	0: Profile generator output 1: External speed	0

Parameter	Description	Units	Range	Default
139	External speed reference	2^{32} counts / rev encoder over 250µs or User Units/s	-715827883 to 715827883	0
140	Output Speed Clamp	2^{32} counts / rev encoder over 250µs or User Units/s	0 to 715827883	53687091
141	Speed Feed Forward Gain	NA	0 to 2000	1000
142	Total profile position reference, [117] + [157].	Encoder counts or User Units	-2^{31} to 2^{31}-1	0
143	Total profile speed reference, [118] + [158].	2^{32} counts / rev encoder over 250µs or User Units/s	-715827883 to 715827883	0
150	Offset Profile Acceleration Rate	2^{32} counts / rev encoder over 250µs / 250µs or User Units/s/s	0 to 4473924	22369
151	Offset Profile Deceleration Rate	2^{32} counts / rev encoder over 250µs / 250µs or User Units/s/s	0 to 4473924	22369
153	Offset Profile Max Speed Clamp	2^{32} counts / rev encoder over 250µs or User Units/s	0 to 715827883	48318382
155	Offset Profile Input Position	Encoder counts or User Units	-2^{31} to 2^{31}-1	0
156	Offset Profile Input Speed	2^{32} counts / rev encoder over 250µs or User Units/s	-2^{31} to 2^{31}-1	0
157	Offset Profile Output Position	Encoder counts or User Units	-2^{31} to 2^{31}-1	0
158	Offset Profile Output Speed	2^{32} counts / rev encoder over 250µs or User Units/s	-2^{31} to 2^{31}-1	0
159	Offset Profile Output Acceleration	2^{32} counts / rev encoder over 250µs / 250µs or User Units/s/s	-2^{31} to 2^{31}-1	0
160	At Offset Target Position	N/A	0: Not at target position 1: At target position	0
161	At Offset Target Speed	N/A	0: Not at target speed 1: At target speed	0
170	Internal position setpoint. Allways in encoder counts	Encoder Counts only	-2^{31} to 2^{31}-1	0
171	Internal position setpoint remainder	N/A	±([97] -1)	0
172	Position setpoint remainder	N/A	±([96] -1)	0
173	Internal offset position setpoint. Allways in encoder counts.	Encoder Counts only	-2^{31} to 2^{31}-1	0
174	Internal offset position remainder	N/A	±([97] -1)	0

Parameter	Description	Units	Range	Default
175	Offset position remainder	N/A	±([96] -1)	0
176	Internal stop position. Allways in encoder counts	Encoder Counts only	-2^{31} to $2^{31}-1$	0
178	Stop position remainder	N/A	±([96] -1)	0
180	Internal profile input position. Allways in encoder counts.	Encoder Counts only	-2^{31} to $2^{31}-1$	0
181	Internal profile output position. Allways in encoder counts.	Encoder Counts only	-2^{31} to $2^{31}-1$	0
182	Internal offset profile input position. Allways in encoder counts.	Encoder Counts only	-2^{31} to $2^{31}-1$	0
184	Internal offset profile input position. Allways in encoder counts.	Encoder Counts only	-2^{31} to $2^{31}-1$	0

11.3 SM-Applications Virtual Parameters

Parameter	Description	Units	Range	Default
90.01	Feedback encoder position	Encoder counts	-2^{31} to 2^{31}-1	NA
90.02	Feedback encoder rev count	rpm	0 to 65535	NA
90.03	Reference encoder position	Encoder counts	-2^{31} to 2^{31}-1	NA
90.04	Reference encoder rev count	rpm	0 to 65535	NA
90.18	Feedback encoder freeze flag	NA	0 : No freeze pulse 1: Freeze pulse	0
90.19	Feedback encoder freeze position	Encoder counts	-2^{31} to 2^{31}-1	NA
90.20	Feedback encoder freeze rev count	rpm	0 to 65535	NA
90.21	Disable drive encoder position check	NA	0: Enable 1: Disable	0
90.22	Drive encoder comms transmit register	NA	0 to 65535	NA
90.23	Drive encoder comms receive register	NA	0 to 65535	NA
90.25	Feedback encoder marker position	Encoder counts	-2^{31} to 2^{31}-1	NA
90.26	Feedback encoder marker rev count	rpm	0 to 65535	NA
90.28	Reference encoder freeze flag	NA	0: No freeze pulse 1: Freeze pulse	0
90.29	Reference encoder freeze position	Encoder counts	-2^{31} to 2^{31}-1	NA
90.30	Reference encoder freeze rev count	rpm	0 to 65535	NA
90.31	Feedback encoder turns and coarse position	Turns and encoder counts	-2^{31} to 2^{31}-1	NA
90.32	Reference encoder turns and coarse position	Turns and encoder counts	-2^{31} to 2^{31}-1	NA
90.33	Feedback encoder freeze turns and coarse position	Turns and encoder counts	-2^{31} to 2^{31}-1	NA
90.34	Reference encoder freeze turns and coarse position	Turns and encoder counts	-2^{31} to 2^{31}-1	NA
90.35	Reference encoder marker position	Encoder counts	-2^{31} to 2^{31}-1	NA
90.36	Reference encoder marker rev count	rpm	0 to 65535	NA
90.37	Feedback encoder marker turns and coarse position	Turns and encoder counts	-2^{31} to 2^{31}-1	NA
90.38	Reference encoder marker turns and coarse position	Turns and encoder counts	-2^{31} to 2^{31}-1	NA
90.41	Reference encoder marker flag	NA	0: No marker pulse 1: Marker pulse	0
90.42	Feedback encoder marker flag	NA	0: No marker pulse 1: Marker pulse	0

Parameter	Description	Units	Range	Default
90.43	Reference encoder source	NA	0: Drive encoder input 1: Slot 1 2: Slot 2 3: Slot 3 4: User Program 5: None	0
90.44	Feedback encoder source	NA	0: Drive encoder input 1: Slot 1 2: Slot 2 3: Slot 3 4: User Program 5: None	0
90.45	Reference marker flag enable	NA	0: Disable marker flag 1: Enable marker flag	0
90.46	Feedback marker flag enable	NA	0: Disable marker flag 1: Enable marker flag	0
90.47	Reference freeze enable	NA	0: Disable freeze 1: Enable freeze	0
90.48	Feedback freeze enable	NA	0: Disable freeze 1: Enable freeze	0

91.01	Shortcut enable	NA	0 to 255	0
91.02	Speed setpoint	0.001 RPM Closed Loop 0.1Hz Open Loop	-2^{31} to 2^{31}-1	0
91.03	Hard speed reference	0.001 RPM	-2^{31} to 2^{31}-1	0
91.05	Maximum speed clamp	RPM Hz	-2^{31} to 2^{31}-1	0
91.06	Speed feedback	0.01 RPM	-2^{31} to 2^{31}-1	NA
91.17	Number of valid CT-Sync messages	NA	-2^{31} to 2^{31}-1	0
91.18	Number of invalid CT-Sync messages	NA	-2^{31} to 2^{31}-1	0
91.19	Number of missing CT-Sync messages	NA	-2^{31} to 2^{31}-1	0
91.20	Synchronisation signal too short	NA	-2^{31} to 2^{31}-1	0
91.21	Inter-option module synchronisation control	NA	0 to 255	0
91.22	Inter-option module synchronisation status	NA	0 to 255	NA

11.4 CTSync Command Reference

Command	Range	
= CTSYNCSetMasterReferences (Reference1%, Reference2%, Reference3%)	Reference1%: Reference2%: Reference3%:	-2^{31} to 2^{31}-1 -2^{31} to 2^{31}-1 0 to 2^{8}-1
= CTSYNCGetSlaveReferences ()	NA	NA

11.5 APC Definitions

The definitions in the following tables are aliases to the values used by the APC commands or are aliases to parameters in the APC. Aliases give meaningful names to numeric values. A user program can either use the numeric value or the alias name. Examples of using aliases are given below.

```
    // Enable APC
    RunModeStatus% = APCSetRunMode(APC_ENABLE)
```

```
//read Position Controller status
PosCtrlStatus% = APCReadPar(APC_POS_CTRL_EN)

IF (PosCtrlStatus% = APC_DISABLE) THEN
    // Position controller enabled but not running
    // place code here
ENDIF
```

11.5.1 Control and Access Functions

Definition	Value	Related Command
APC_OFF	0	APCSetRunMode
APC_DISABLE	1	
APC_ENABLE	2	

11.5.2 Feedback and Reference Encoder

Definition	Value	Related Command
APC_DRIVE_ENC	0	APCSetReferenceSource APCSetFeedbackSource APCGetReferenceStatus APCGetFeedbackStatus
APC_SLOT1_ENC	1	
APC_SLOT2_ENC	2	
APC_SLOT3_ENC	3	
APC_USER_ENC	4	

11.5.3 References

Definition	Value	Related Command
APC_LINEAR_CAM	0	APCSetCAMInterpolationMode
APC_COSINE_CAM	1	
APC_ARRAY_CAM	-1	
APC_UNLOCKED	0	APCSetDigLockMode
APC_LOCKED	1	
APC_NEVER_LOCK	2	
APC_PROF_STOP	0	APCSetStopMode
APC_FAST_STOP	1	
APC_STOP_REF	0	APCSelectReference
APC_POSITION_REF	1	
APC_SPEED_REF	2	
APC_CAM_REF	3	
APC_DIG_LOCK_REF	4	
APC_NO_ACT_FRZ	0	APCSelectActionOnFreeze
APC_DIG_LOCK_FRZ	1	
APC_CAM_ON_FRZ	2	

11.5.4 Offset Profile Generator

Definition	Value	Related Command
APC_OFF_DISABLED	0	APCSelectOffset
APC_OFF_POS	1	
APC_OFF_SPD	2	

11.5.5 Position Loop

Definition	Value	Related Command
APC_HARD_SPD_REF	0	APCSetupOutputChannel
APC_SPEED_REF	2	
APC_PARAMETER	3	
APC_USER_REF	4	
APC_QUIET	5	
APC_HARD_SPD_REF_I	10	(Available in Sypt Pro V2.0.11)
APC_SPEED_REF_I	12	(Available in Sypt Pro V2.0.11)
APC_PARAMETER_I	13	(Available in Sypt Pro V2.0.11)
APC_USER_REF_I	14	(Available in Sypt Pro V2.0.11)

11.5.6 Operation Status Definitions

Definition	Value	Related Command
APC_FAIL	0	Generic status definition
APC_OK	1	
APC_ACCESS_DENY	-3	
APC_INVALID_ARGU	-4	

11.5.7 APC and CTSync Output Channel Definitions

Definition	Value	Related Command
CHAN_SUCCESS	1	Generic output channel definition
CHAN_NOT_SET	0	
CHAN_INVALID	-1	
CHAN_NOT_MAST	-2	
CHAN_ACCESS_DENY	-3	
CHAN_INVALID_PAR	-4	
CHAN_INVALID_OP	-5	

11.5.8 CTSync Definitions

Definition	Value	Related Command
USR_VALUE_OK	1	Generic CTSync definition
USR_VAL_UNCONFD	0	
USR_INVALID_CHAN	-1	
USR_NOT_MASTER	-2	
USR_ACCESS_DENY	-3	
USR_PAR_INVALID	-4	
USR_INVALID	-5	

11.5.9 APC Parameter Definitions

Definition	Value
APC_POS_CTRL_EN	0
APC_ABS_MODE	1
APC_POS_RST_OFF	2
APC_SRC_RST_DIS	3
APC_ENABLE_STATUS (SyptPro V02.0.11)	4
APC_RESET_STATUS (SyptPro V02.0.11)	5
APC_REF_POS_RST_OFF (SyptPro V02.0.11)	6
APC_SAMPLE_TIME	13
APC_NO_TURN_BITS	14
APC_VERSION_NUM	15
APC_RFSRC_POS	20
APC_RFSRC_FRZPOS	21
APC_RFSRC_FRZFLG	22
APC_RFSRC_MRKPOS	23
APC_RFSRC_MRKFLG	24
APC_RFSRC_MRKACT	25
APC_RFSRC_SPD	26
APC_RFSRC_INV	27
APC_RF_POS	28
APC_RF_FRZPOS	29
APC_RF_MRKPOS	30

Definition	Value
APC_REF_IN	31
APC_REF_IN_DIS	32
APC_FBSRC_POS	40
APC_FBSRC_FRZPOS	41
APC_FBSRC_FRZFLG	42
APC_FBSRC_MRKPOS	43
APC_FBSRC_MRKFLG	44
APC_FBSRC_MRKACT	45
APC_FBSRC_SPD	46
APC_FBSRC_INV	47
APC_FB_POS	48
APC_FB_FRZPOS	49
APC_FB_MRKPOS	50
APC_FBCK_IN	51
APC_FBCK_IN_DIS	52
APC_RIGID_LOCK	60
APC_DIGLCK_MODE	61
APC_DIGLCK_NUM	62
APC_DIGLCK_DEN	63
APC_DIGLCK_SPD	64
APC_DIGLCK_POS	65
APC_CAM_STRT_IDX	74
APC_CAM_SIZE	75
APC_CAM_SEG_LIM	76
APC_CAM_INT_MODE	77
APC_CAM_SGL_SHOT	78
APC_CAM_ABS_RST	79
APC_CAM_OUT_NUM	80
APC_CAM_OUT_DEN	81
APC_CAM_INDEX	82
APC_CAM_POS_SEG	83
APC_CAM_ARR_SIZE	84
APC_CAM_RST_IDX	85
APC_CAM_RST_POS	86
APC_CAM_SS_COMPLETE	87
APC_CAM_SS_FRZ_REARM	88
APC_CAM_WRAP_THIS_CYCLE	89
APC_STOP_MODE	90
APC_STOP_POS	91
APC_POS_SETPOINT	92
APC_POS_OFFSET	93
APC_SPD_SETPOINT	94

Definition	Value
APC_SPEED_OFFSET	95
APC_USER_CONV_N	96
APC_USER_CONV_D	97
APC_REF_SELECT	100
APC_ACT_ON_FRZ	101
APC_OFF_SELECT	102
APC_ACCEL_RAMP	110
APC_DECEL_RAMP	111
APC_PRF_MAX_SPD	113
APC_PROF_DISABLE	114
APC_PROF_IN_POS	115
APC_PROF_IN_SPD	116
APC_PROF_OUT_POS	117
APC_PROF_OUT_SPD	118
APC_PROF_OUT_ACC	119
APC_AT_POS	120
APC_AT_SPD	121
APC_POS_ERROR	130
APC_KV	131
APC_PGAIN_CLAMP	132
APC_OUTPUT_NUM	133
APC_OUTPUT_DEN	134
APC_OUTPUT_SPEED	135
APC_EXTREFPOSSEL	136
APC_EXT_REF_POS	137
APC_EXTREFSPDSEL	138
APC_EXT_REF_SPD	139
APC_OUTSPD_CLAMP	140
APC_SPD_FF_CLAMP	141
APC_TOT_POS_REF	142
APC_TOT_SPD_REF	143
APC_OFF_ACCEL	150
APC_OFF_DECEL	151
APC_OFF_S_CLAMP	153
APC_OFF_IN_POS	155
APC_OFF_IN_SPD	156
APC_OFF_OUT_POS	157
APC_OFF_OUT_SPD	158
APC_OFF_OUT_ACC	159
APC_OFF_AT_POS	160
APC_OFF_AT_SPD	161
APC_INT_POS_SETPOINT	170

Definition	Value
APC_INT_POS_SETPOINT_REM	171
APC_POS_SETPOINT_REM	172
APC_OFF_INT_POS_SETPOINT	173
APC_OFF_INT_POS_SETPOINT_REM	174
APC_OFF_POS_SETPOINT_REM	175
APC_INT_STOP_POS	176
APC_STOP_REM	178
APC_INT_PROF_IN_POS	180
APC_INT_PROF_OUT_POS	181
APC_OFF_INT_PROF_IN_POS	182
APC_OFF_INT_PROF_OUT_POS	184

11.6 SM-Applications Setup Parameters

Parameters in this section are referred to as #81.xx. They are aliases to menu 15, 16 or 17, depending on which slot the SM-Applications module is fitted to. When writing code for the SM-Applications module, using #81.xx ensures that the module can be fitted to any slot, and the program will still run.

Table 11-1

Parameter	Description	Units	Range	Default
81.06	RS485 port mode	NA	25 CTSync Master 26 CTSync Slave 0 to 255	1
81.12	POS Task scheduling rate	Milliseconds	0: Disable 1: 0.25ms 2: 0.5ms 3: 1ms 4: 2ms 5: 4ms 6: 8ms	0
81.13	Auto-run enable	NA	0: Disable 1: Enable	0
81.16	Encoder data update rate (APC and Menu 90)	NA	0: Every 0.25ms 1: Every POS Task 2: Every Clock Task 3: Never	0
81.38	Disable drive trip on APC Runtime Error (prevents Tr81 if required)	NA	0: Disable 1: Enable	0
81.42	Pass digital input 0 through to the freeze counters	NA	0: Disable 1: Enable	0

11.7 Conversion Factors

11.7.1 The following table shows how to convert values between units.

Table 11-2

Convert From	Convert To	Multiply By
Angstrom	inch (in.) - US	$3.937*10^{-9}$
centimetre (cm)	inch (in.) - US	0.3937
foot (ft)	metre	0.3048
inch (in.) - US	2.5400058	centimetre (cm)
inch (in.) - UK	inch (in.) - US	0.9999972
metre (m)	Angstrom	10^{10}
metre (m)	foot (ft)	3.280833
metre (m)	inch (in.)	39.37
metre (m)	yard (yd)	1.09361
metre (m)	mile (US statute)	$6.2137 * 10^{-4}$
yard (yd)	metre (m)	0.91440
mile (US statute)	foot (ft)	5280
radian (rad)	degree (deg)	57.29578
feet per second	miles per hour	0.68182
metres per second	miles per hour	2.23693
rpm	radians per second	0.10472
miles per hour	centimetres per second	44.7041
miles per hour	feet per second	1.4667

11.8 Documentation

A list of other documents or manuals that may be referred to is given below:

Document name	Part number
SM-Applications User Guide	0471-0007-xx
SM-Applications Lite User Guide	0471-0030-xx
Unidrive SP User Guide	0471-0000-xx
Unidrive SP Advanced User Guide	0471-0002-xx

11.9 Overview

Figure 11-1 Reference Source Position Counter

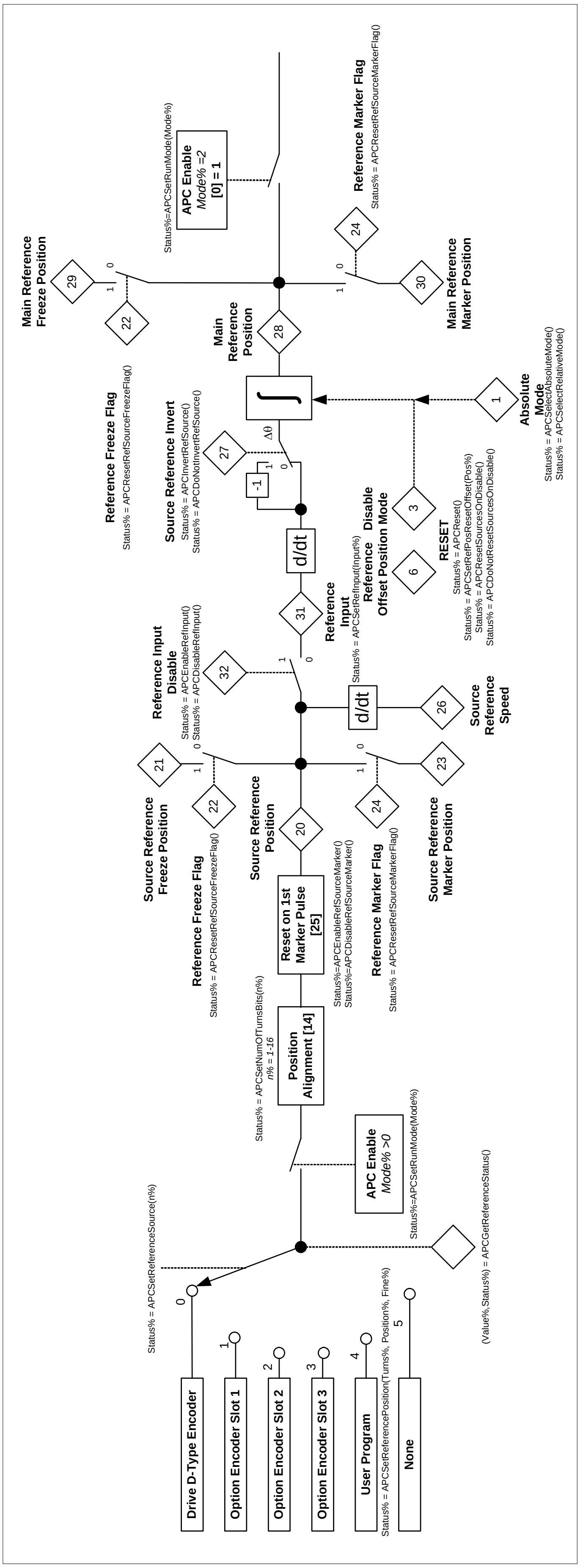


Figure 11-2 Feedback Source Position Counter

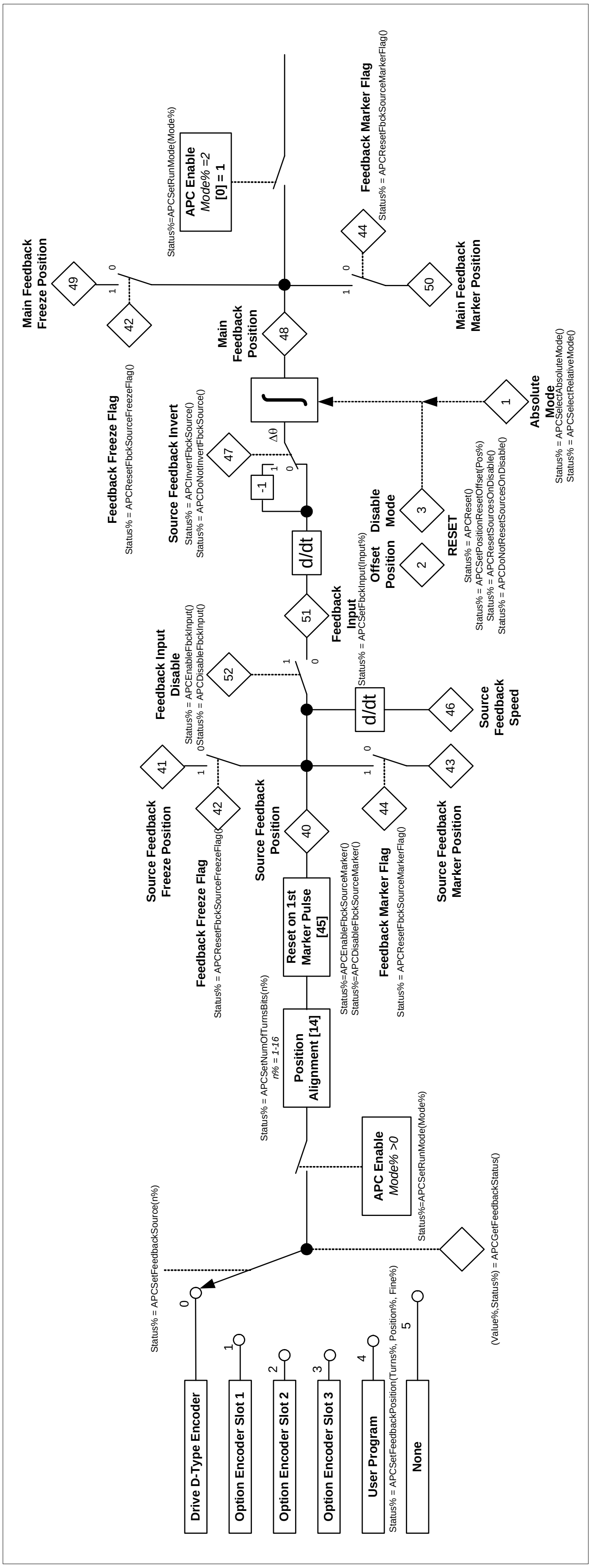


Figure 11-3 APC References

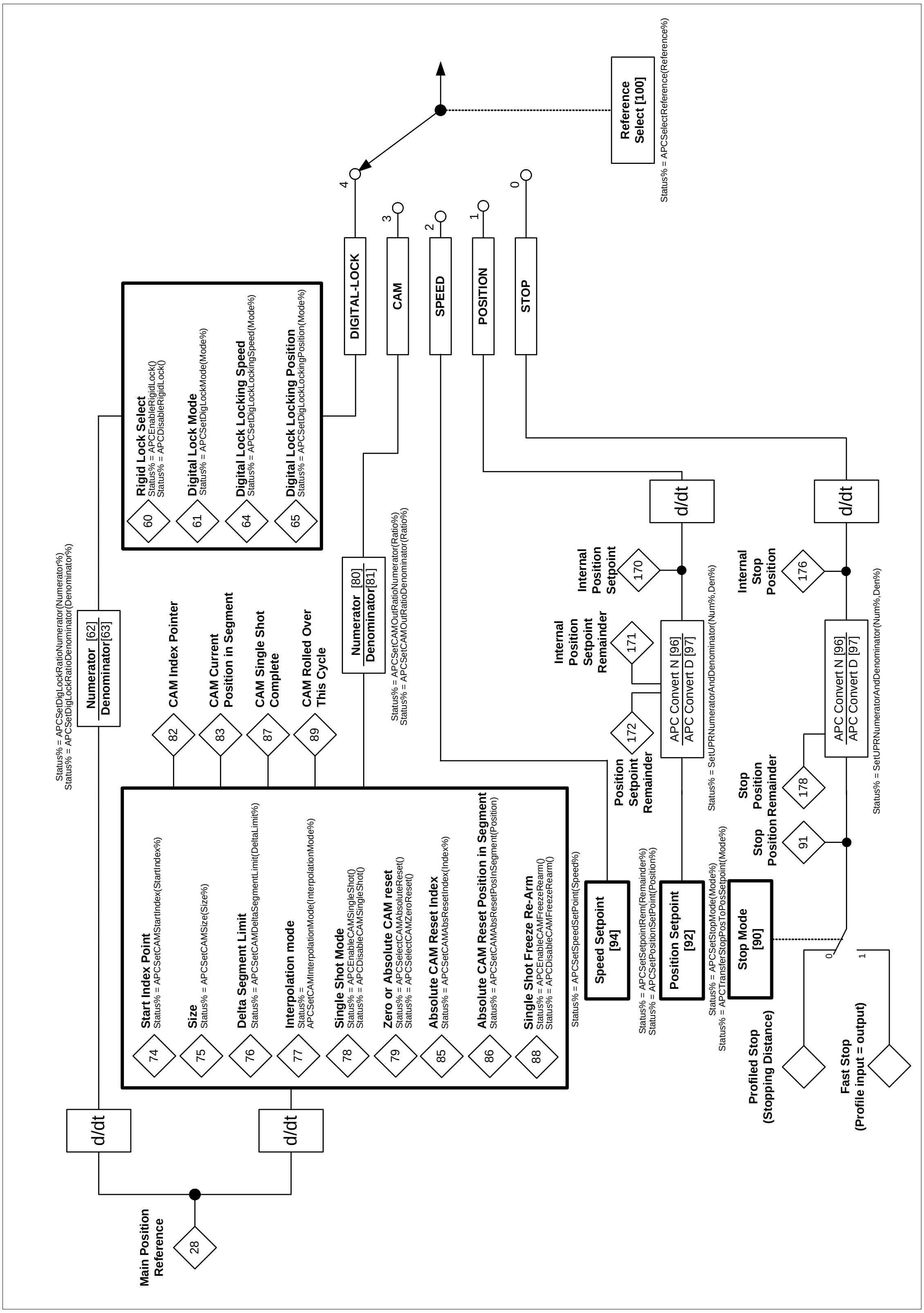


Figure 11-4 Profile Generator

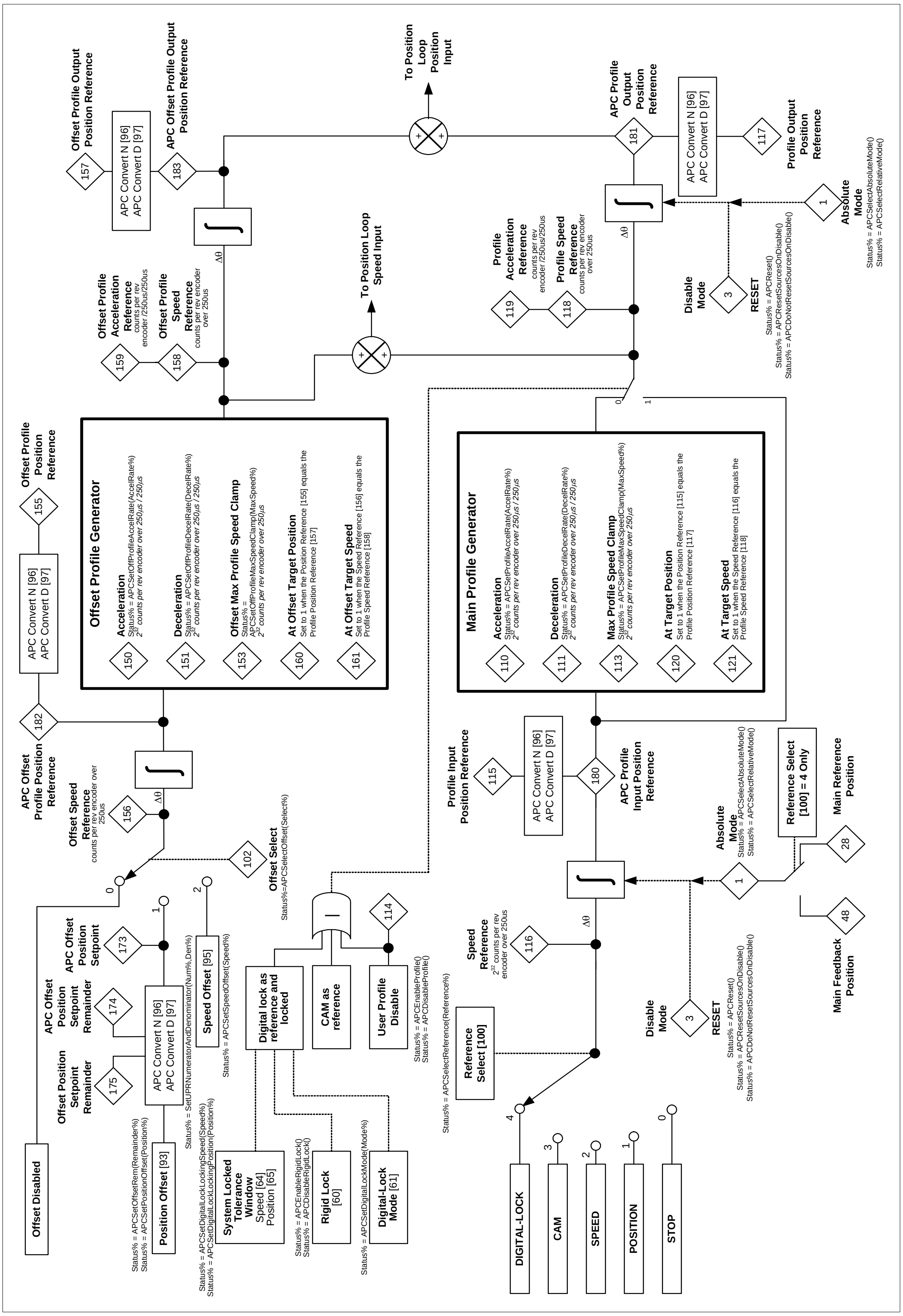
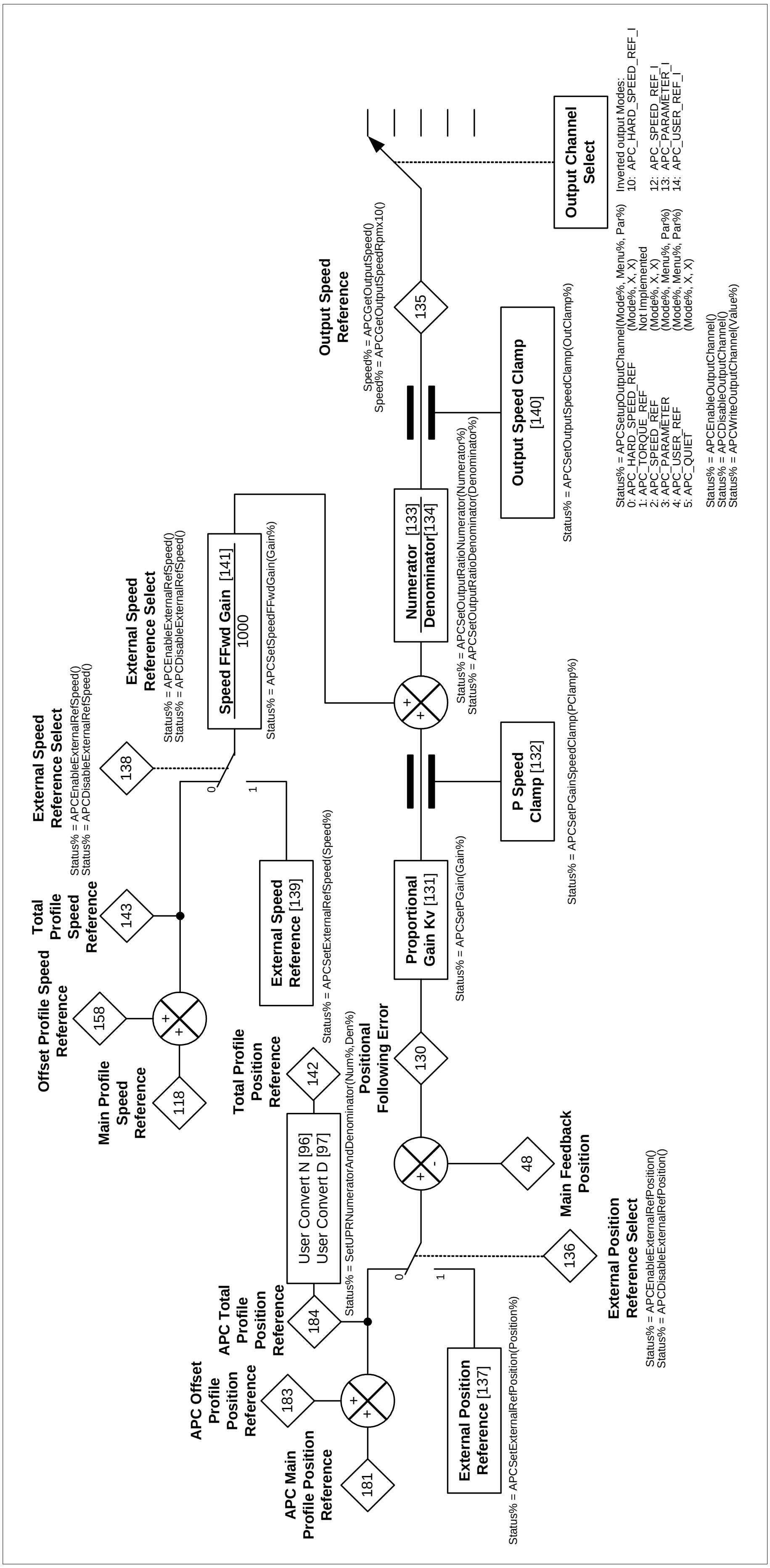


Figure 11-5 Position Loop



Index

0471-0034-05